



WEB

## 第 4 章 客户端动态脚本 JavaScript

# 本章目录

- 4.1 JavaScript简介
- 4.2 JavaScript开发工具
- 4.3 JavaScript的数据类型
- 4.4 JavaScript的运算符
- 4.5 JavaScript的对象
- 4.6 JavaScript的函数
- 4.7 JavaScript的流程控制
- 4.8 JavaScript事件驱动
- 4.9 JavaScript事件处理

# 引例

必填信息

|      |                          |
|------|--------------------------|
| 用户名  | <input type="text"/>     |
| 密码   | <input type="password"/> |
| 确认密码 | <input type="password"/> |
| 电子邮箱 | <input type="text"/>     |

请认真填写, 以便日后你取回丢失的密码

头像



选填信息

|      |                                      |
|------|--------------------------------------|
| QQ号码 | <input type="text"/>                 |
| 个人主页 | <input type="text" value="http://"/> |

确定

重置

返回

图 4-1

# 引例

- 我们在浏览 Internet 上各式各样的页面过程中，经常会遇到一些“动态”效果，这里的“动态”不是指 Flash、GIF 动画，而是指由于我们对键盘或鼠标的操作产生的交互效果，如图 4-1 所示弹出的警告框。
- 类似以上这种“动态”效果其实是采用客户端脚本来实现的。
- 常见的客户端脚本有 Netscape 的 JavaScript 和微软的 VBScript、Jscript 等。

## 4.1 JavaScript 简介

- JavaScript 语言的前身叫做 Livescript。自从 Sun 公司推出著名的 Java 语言之后，Netscape 公司引进了 Sun 公司有关 Java 的程序概念，将自己原有的 Livescript 重新进行设计，并改名为 JavaScript。
- JavaScript 是一种基于对象（ Object Based ）和事件驱动（ Event Driver ）并具有安全性能脚本语言。

# 4.1 JavaScript 简介

- 在标准的 HTML 语言中，通过嵌入 JavaScript 程序，从而使得客户与浏览器交互。
- JavaScript 是动态的，它可以直接对客户输入做出响应，无须经过 Web 服务程序。
- JavaScript 是跨平台的，它依赖于浏览器本身，与操作环境无关，只要能运行浏览器的计算机，并支持 JavaScript 的浏览器就可以正确执行。

# JavaScript 和 Java 的区别

## 1. 解释与执行方式的不同

Java 的源代码在执行之前,必须经过编译,而 JavaScript 是一种解释性编程语言,其源代码在执行前不需经过编译。

## 2. 执行位置的不同

Java 程序一般以 JSP 嵌入代码的形式在服务器端被执行,而 JavaScript 必须由客户端浏览器解释执行。

## 3. 基于对象与面向对象

Java 是一种真正的面向对象的语言,而 JavaScript 是一种脚本语言。

## 4. 弱变量与强变量

Java 中所有变量在编译之前必须做声明,JavaScript 的变量并不需要明确的定义。

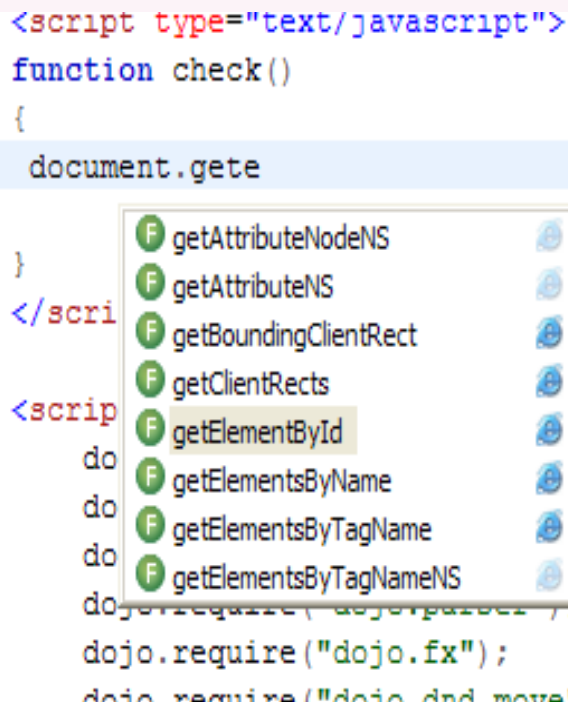
## 4.2 JavaScript 开发工具

### 4.2.1 Aptana 简介

Aptana 是一个基于 Eclipse 平台的开发工具，属于开源软件，该工具是一个功能非常强大的专注于客户端动态程序开发的 Web 开发工具，该工具最主要的特点便是代码智能提示，即支持 CSS、JavaScript、HTML 代码提示，包括 JavaScript 函数和代码语法错误提示。

如图 4- 2 所示：





**getElementById(id: String) : HTMLElement**

Returns the **Element** node that matches the given **ID**.

**Supported:** IE 5.0+, Mozilla 1.0+, Netscape 6.0+, Safari 1.0+, Opera 7.0+

## 4.2.2 Aptana 的安装

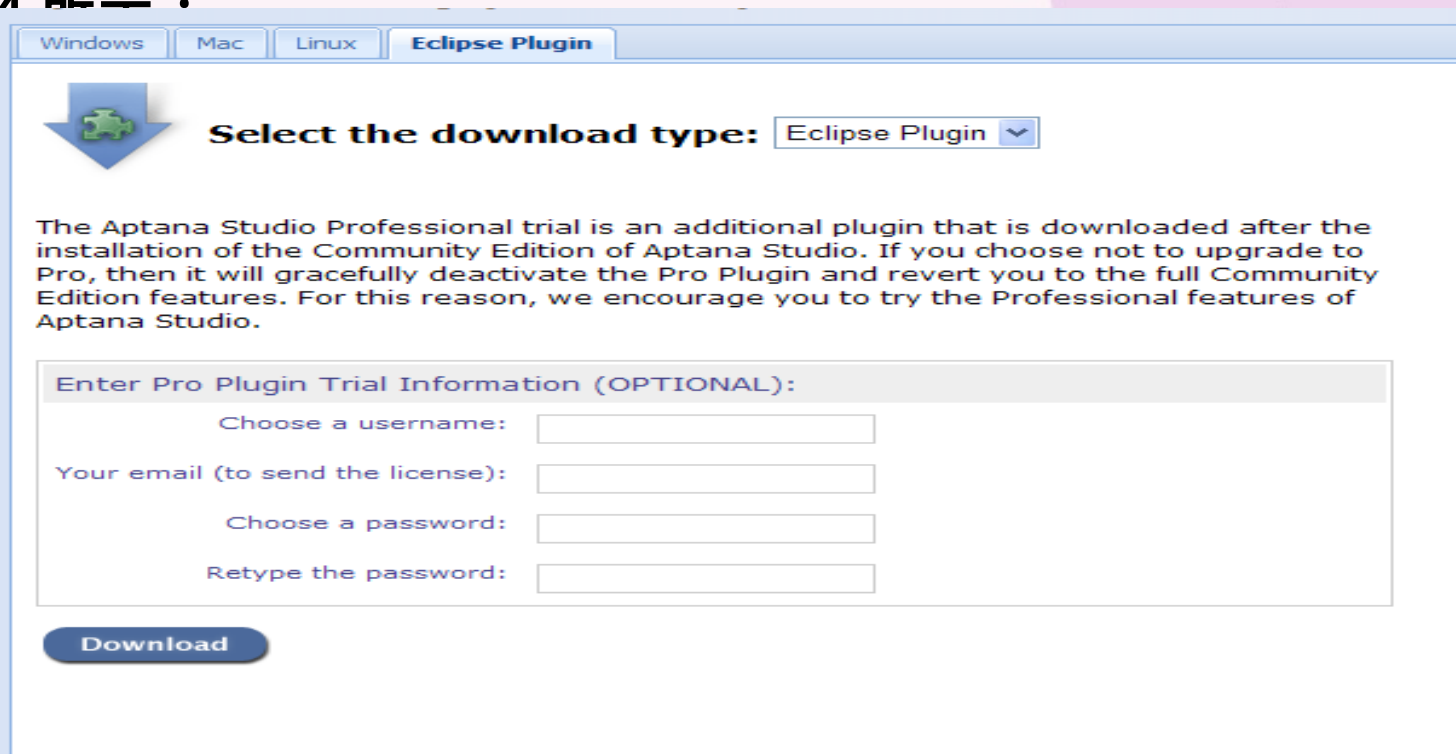
1.首先，打开网址 <http://www.Aptana.com/products/>，如图 4- 3 所示，上面有社区版和专业版的区别，请仔细阅读，以选择合适的开发工具。

| JavaScript Debugging                                      |   |         |
|-----------------------------------------------------------|---|---------|
| Firefox                                                   | ✓ | ✓       |
| Internet Explorer                                         | ✗ | ✓       |
| Advanced Features                                         |   |         |
| JavaScript/Ruby Automation of the Development Environment | ✓ | ✓       |
| Project Reporting Engine                                  | ✗ | ✓       |
| Support                                                   |   |         |
| Priority response on Forums                               | ✗ | ✓       |
| Priority support ticket system                            | ✗ | ✓       |
| Priority voting on new features                           | ✗ | ✓       |
| Access to early builds of all Studio plugins              | ✗ | ✓       |
| SVN access to new features before release                 | ✗ | ✓       |
| download                                                  |   | buy pro |

图 4- 3 Aptana 下载页面截图  
Web 程序设计 第四章

## 4.2.2 Aptana 的安装

2. 确定自己要下载的版本，单击列表下方的【download】按钮，转到下载页面 <http://www.Aptana.com/download/>，在这里可以选择下载单独的工具，还是插件包，这里选择插件包，如图 4-4 所示。



The screenshot shows a web browser window with tabs for Windows, Mac, Linux, and Eclipse Plugin. The main content area has a green puzzle piece icon and a dropdown menu set to 'Eclipse Plugin'. Below this is a text block explaining the trial version. Further down is a section for optional trial information with four input fields: username, email, password, and password confirmation. A 'Download' button is at the bottom.

Windows Mac Linux **Eclipse Plugin**

Select the download type: Eclipse Plugin

The Aptana Studio Professional trial is an additional plugin that is downloaded after the installation of the Community Edition of Aptana Studio. If you choose not to upgrade to Pro, then it will gracefully deactivate the Pro Plugin and revert you to the full Community Edition features. For this reason, we encourage you to try the Professional features of Aptana Studio.

Enter Pro Plugin Trial Information (OPTIONAL):

Choose a username:

Your email (to send the license):

Choose a password:

Retype the password:

Download

图 4-4 选择下载类型

## 4.2.2 Aptana 的安装

选择【Eclipse Plugin】选项卡。单击【Download】按钮，页面接着会跳转到 <http://update.Aptana.com/update/3.2/> 页面。如图 4- 5 所示：

This site is designed to be used though the Eclipse or Aptana update manager.

### Installing this Plugin via Aptana or Eclipse

1. From the **Help** menu, select **Software Updates > Find and Install...** to open an **Install/Update** pop-up window.
2. On the **Install/Update** pop-up window, choose the **Search for new features to install** option, and click the **Next** button.
3. Set up a new remote site to scan for updates.
  - a. Click the **New Remote Site...** button to open a **New Update Site** pop-up window.
  - b. On the **New Update Site** pop-up window, type the name of the new plug-in in the site **Name** text box.
  - c. In the **URL** text box, type the URL <http://update.apptana.com/update/3.2/> for the update site.
  - d. Click **OK**.
  - e. Click the **Finish** button to open an **Updates** window.
4. On the **Updates** window, check the box next to the name of the plug-in, and click the **Next** button.
5. Choose the option to accept the terms of the license agreement, and click the **Next** button.
6. Click the **Finish** button.
7. Click the **Install All** button.

### Manual Installation



Download Plugin Update Site

图 4- 5 下载页面  
Web 程序设计 第四章

## 4.2.2 Aptana 的安装

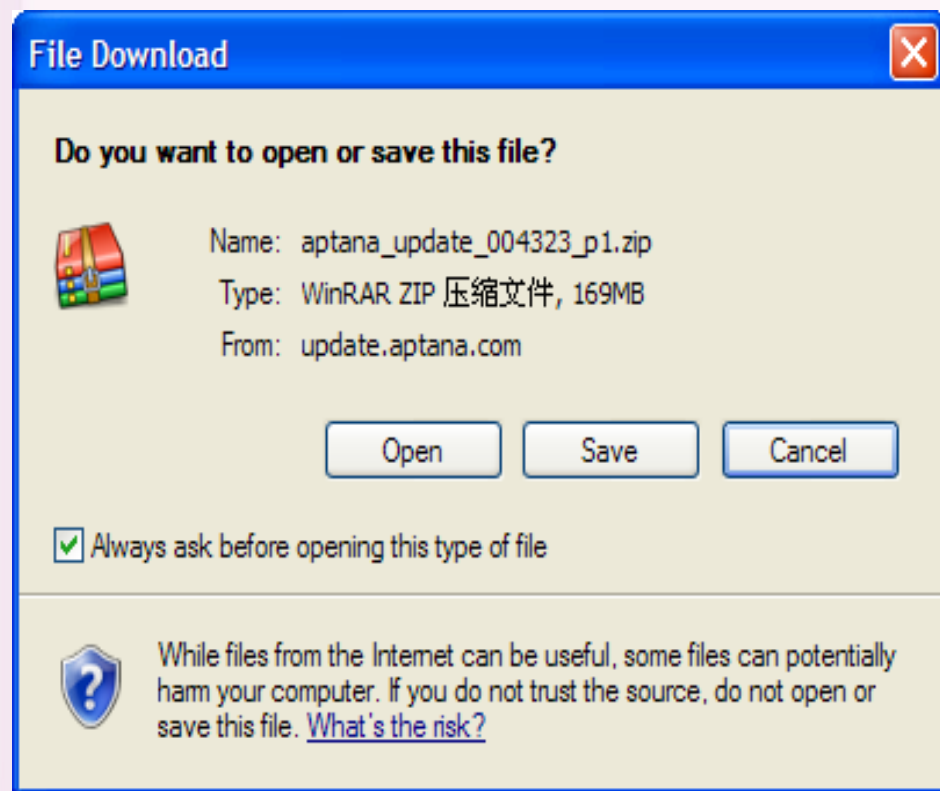


图 4-6 操作（运行\保存）选择对话框

该网页上有如何进行在线安装和离线安装的办法，本书以离线安装为例，单击“Down Plugin Update Site”图标下载，弹出如图 4- 6 所示对话框

单击【 Save 】按钮，将文件保存到本地，下载完成后得到一个 zip 包，对该包无需做任何处理。

## 4.2.2 Aptana 的安装

3. 通过 Eclipse 的插件安装功能进行 Aptana 的安装。单击【Help】下的【Software Updates】菜单项，如图 4-7 所示：

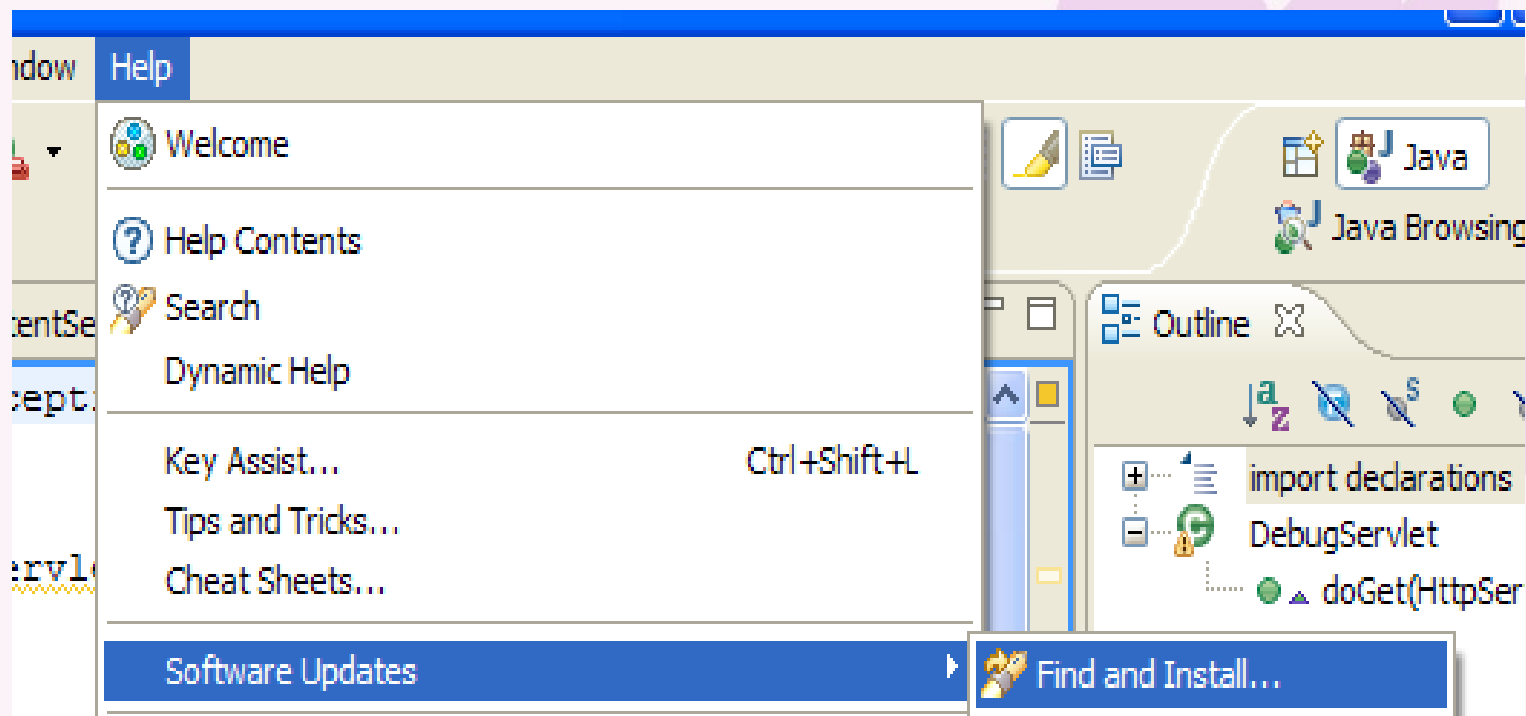


图 4-7 Eclipse 插件安装

## 4.2.2 Aptana 的安装

单击【Find and Install...】菜单项，系统会弹出如图 4-8 所示的界面，供用户选择安装模式：

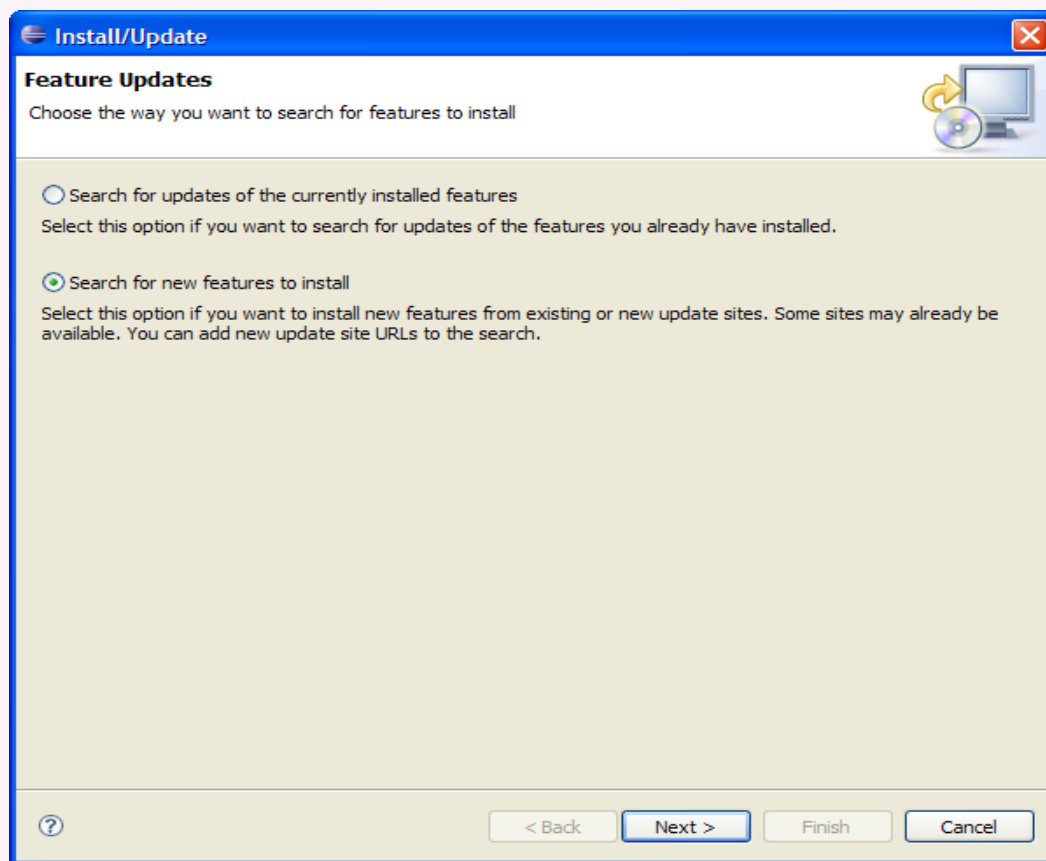


图 4-8 选择安装模式

## 4.2.2 Aptana 的安装

这里选择第二种“Search for new features to install”，第二种是检查已安装插件有没有更新。单击【next】按钮，弹出如图 4- 9 所示的安装选择页面。

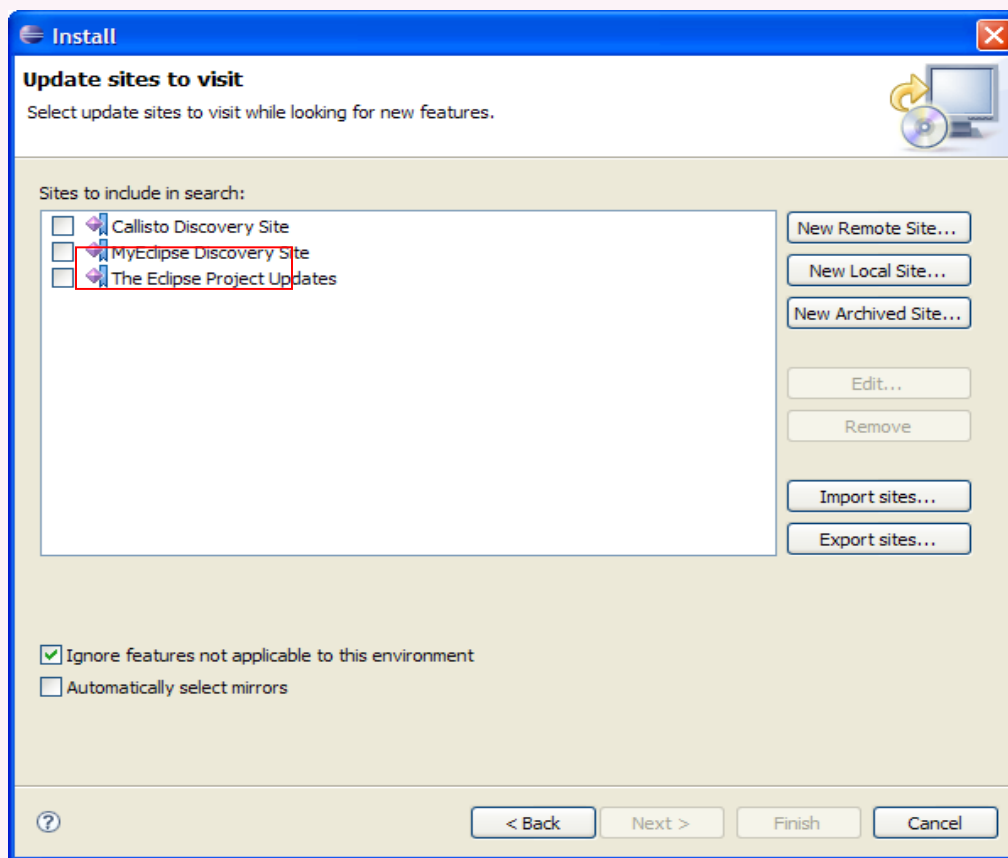


图 4- 9 安装选择界面



## 4.2.2 Aptana 的安装

单击【New  
Archived  
Site....】按钮，  
弹出选择安装文件  
对话框，如图  
4- 10 所示：

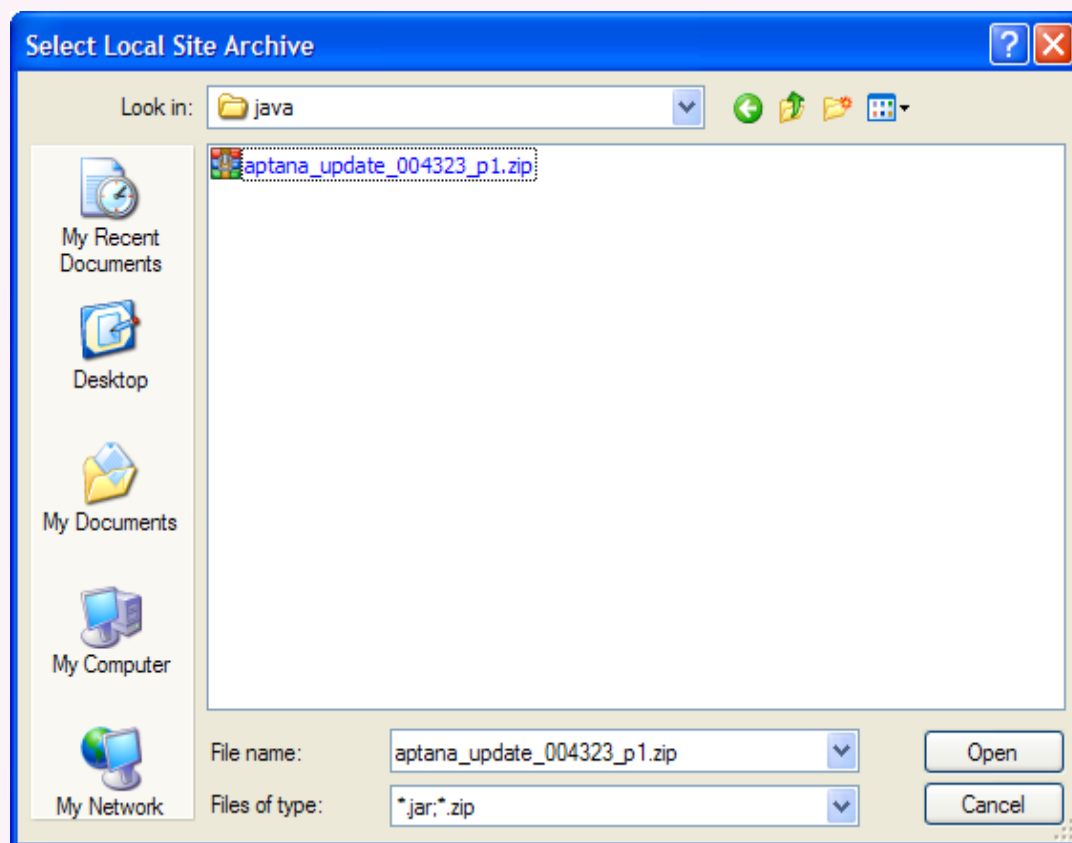


图 4- 10 选择安装文件

## 4.2.2 Aptana 的安装

选择刚才下载的 zip 包，单击【open】按钮，弹出如图 4-11 所示编辑对话框：

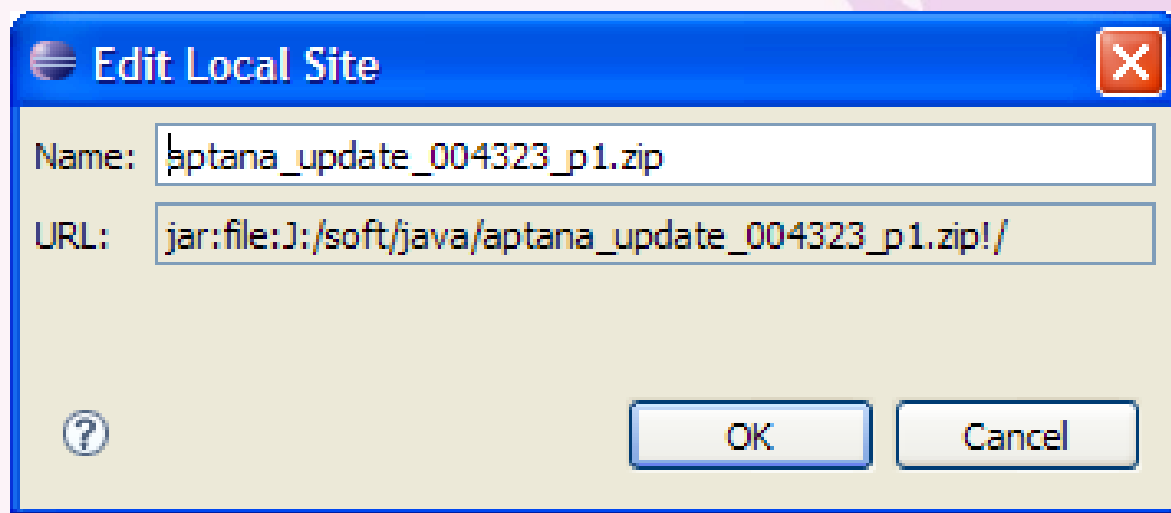


图 4-11 编辑路径

## 4.2.2 Aptana 的安装

单击【OK】  
按钮，返回如  
图 4- 12 所示对  
话框：

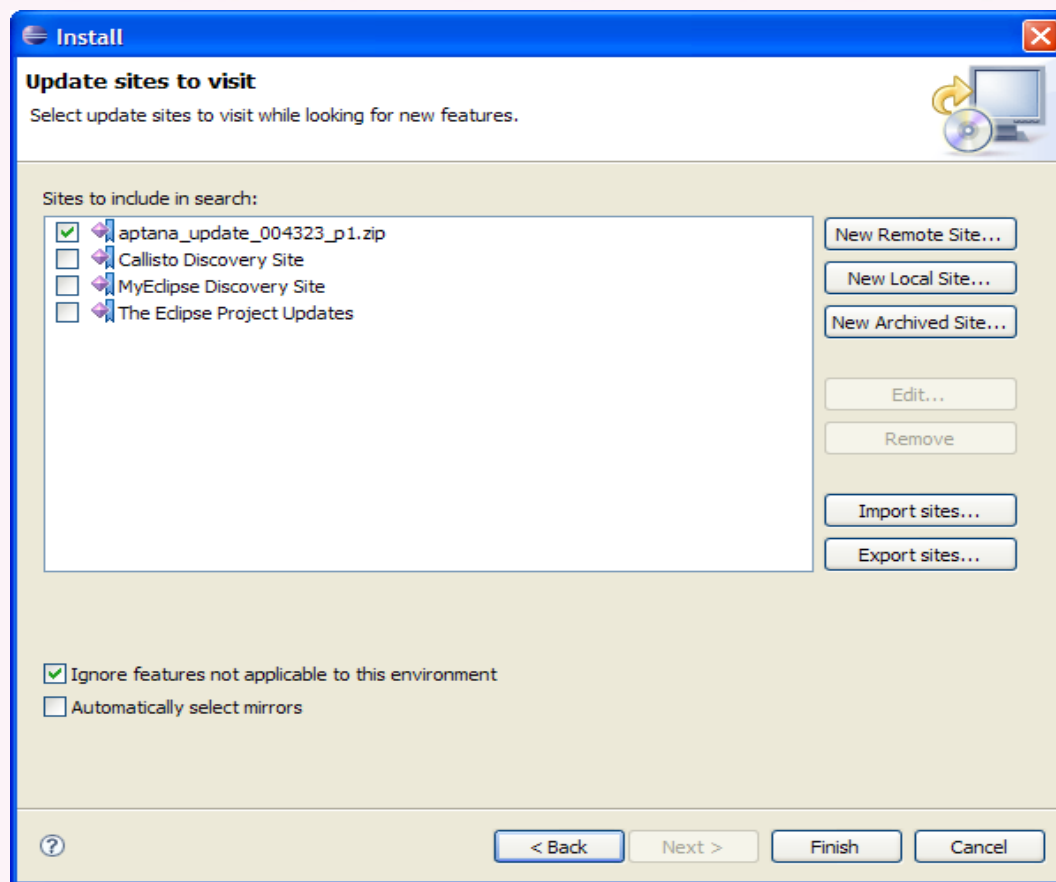


图 4- 12 安装文件添加成功

## 4.2.2 Aptana 的安装

单击

【 Finish 】按钮，弹出如图 4-13 所示对话框：

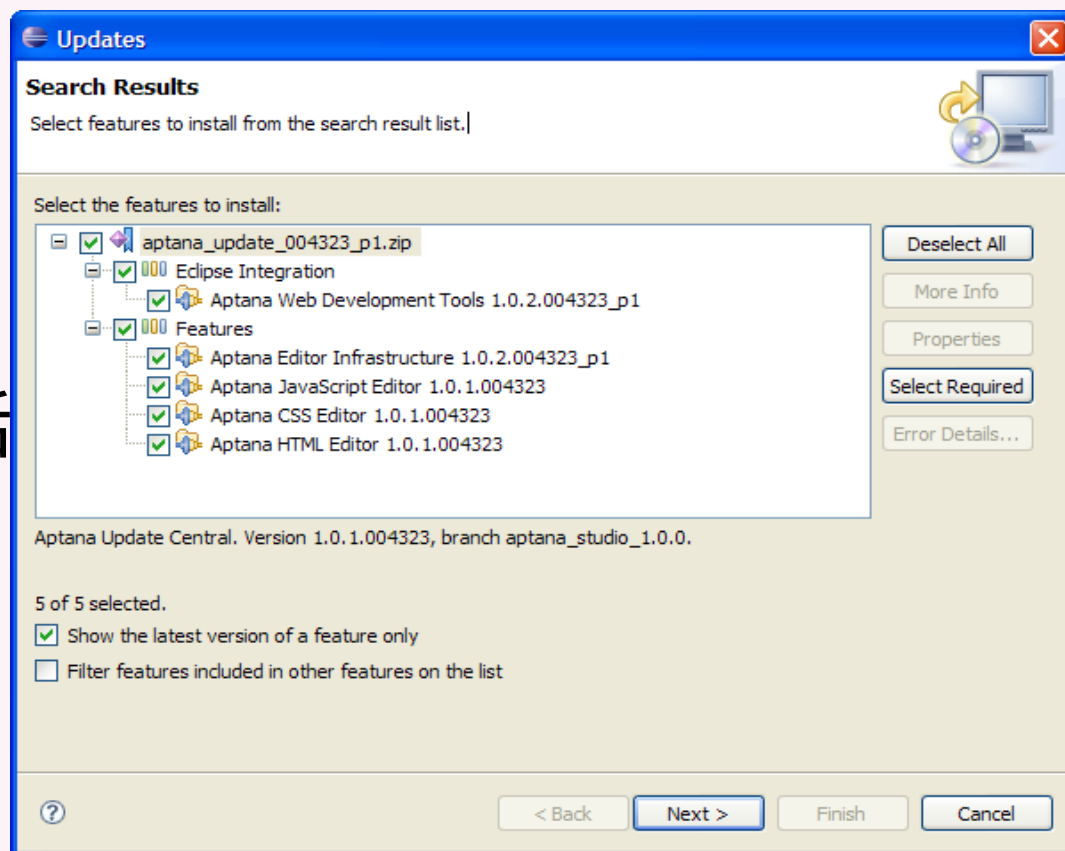


图 4-13 选择需要安装的功能组件

## 4.2.2 Aptana 的安装

把所有选项选上，单击【Next】按钮，弹出如图 4-14 所示许可证协议对话框：

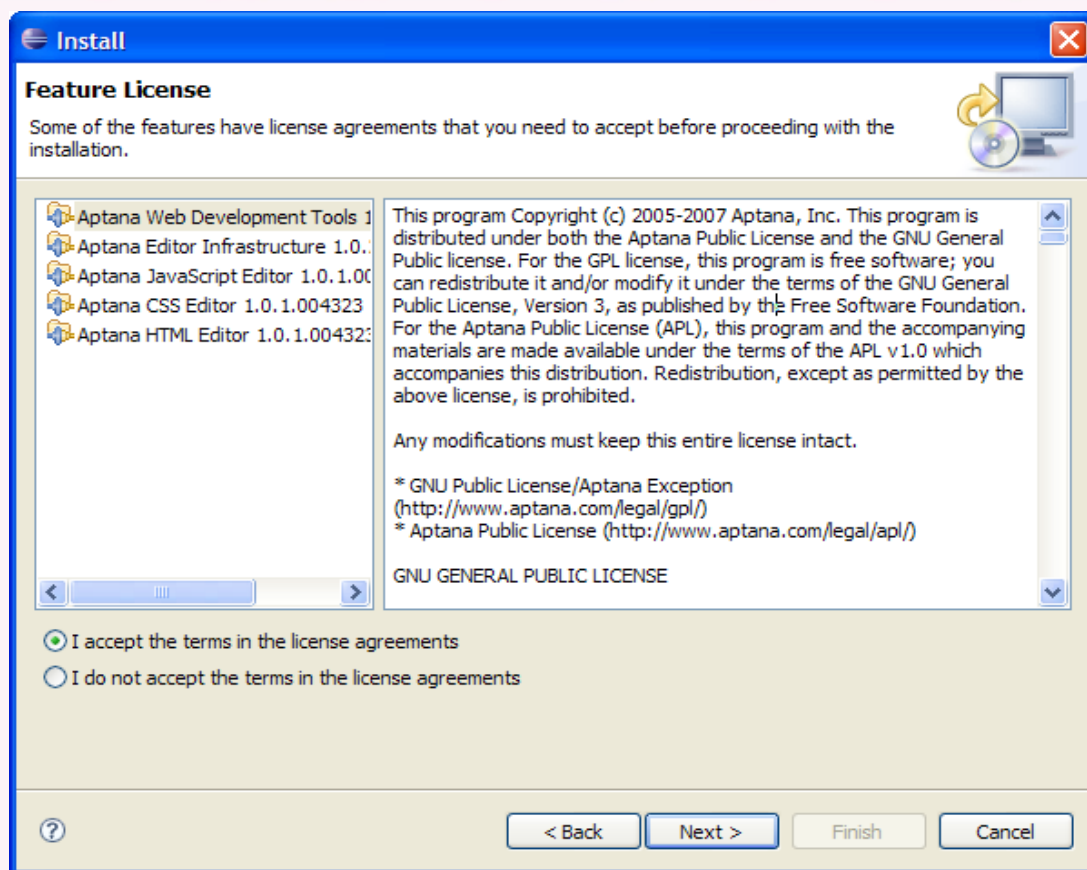


图 4-14 许可证协议对话框

## 4.2.2 Aptana 的安装

接受协议，单击  
【 Next 】按钮  
，弹出如图 4-  
15 所示安装路  
径选择对话框：

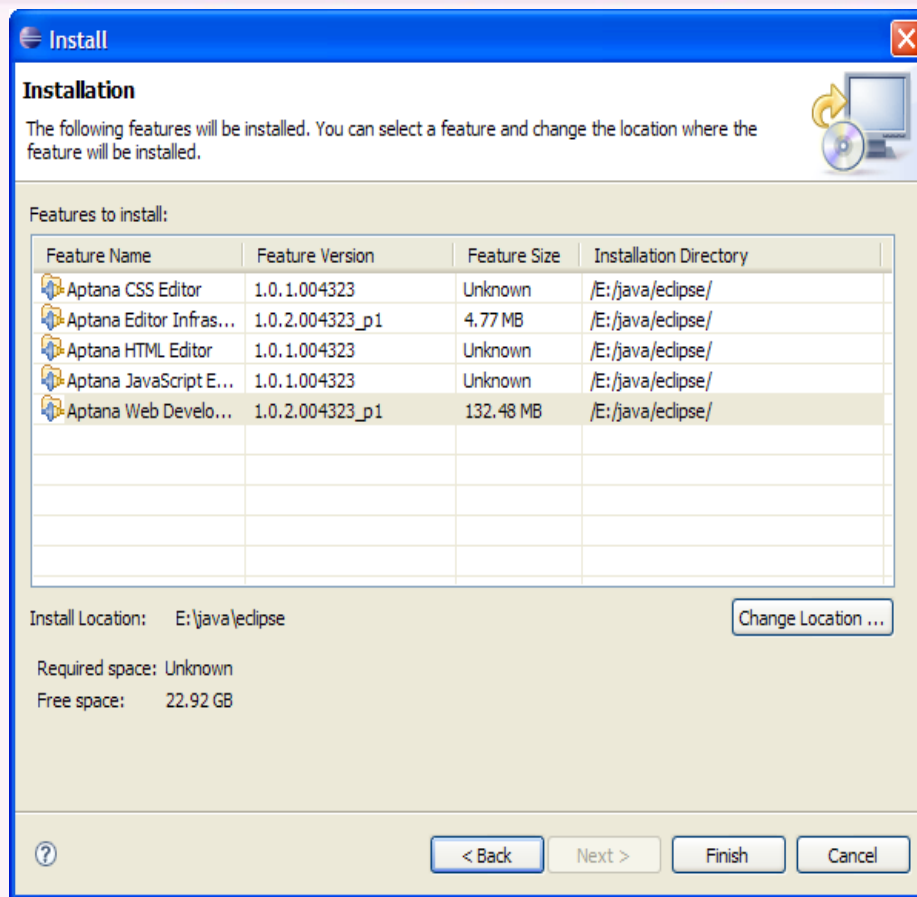


图 4- 15 选择安装路径

## 4.2.2 Aptana 的安装

路径确定后，单击【Finish】按钮。弹出如图 4- 16 所示的插件验证界面：

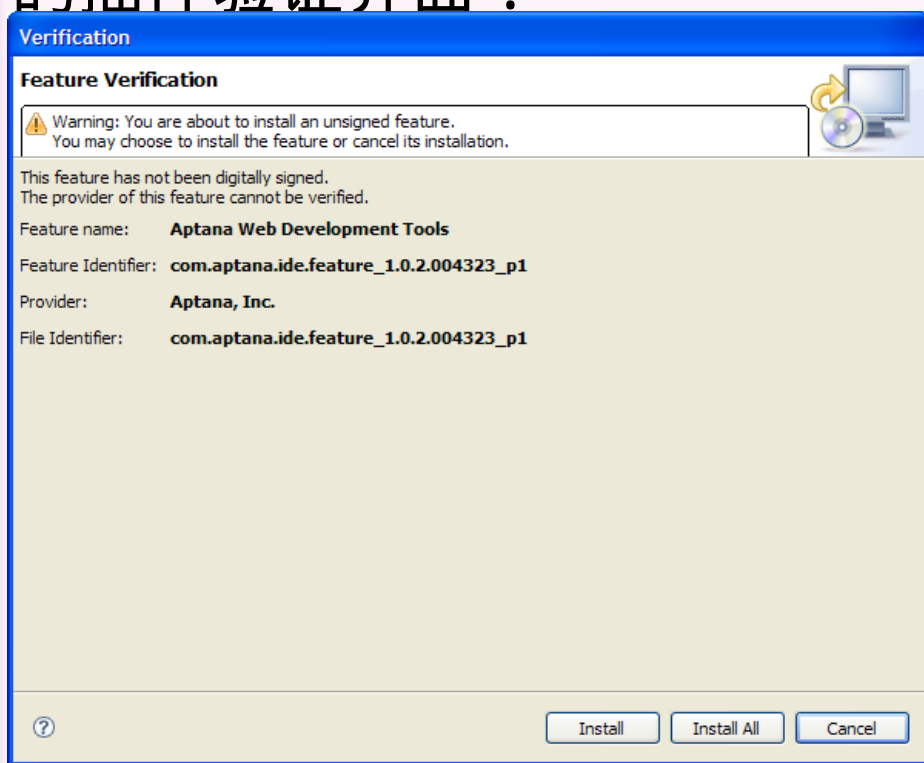


图 4- 16 插件验证界面

插件验证界面，提示 Aptana Web Development Tools 插件没有数字签名，这个不受什么影响，单击【Install All】按钮开始安装。安装完成后，会提示重启 Eclipse。

## 4.2.2 Aptana 的安装



安装如果成功完成，则重启后，出现 Aptana 的欢迎界面。如图 4- 17 所示：

图 4- 17 Aptana 的欢迎界面

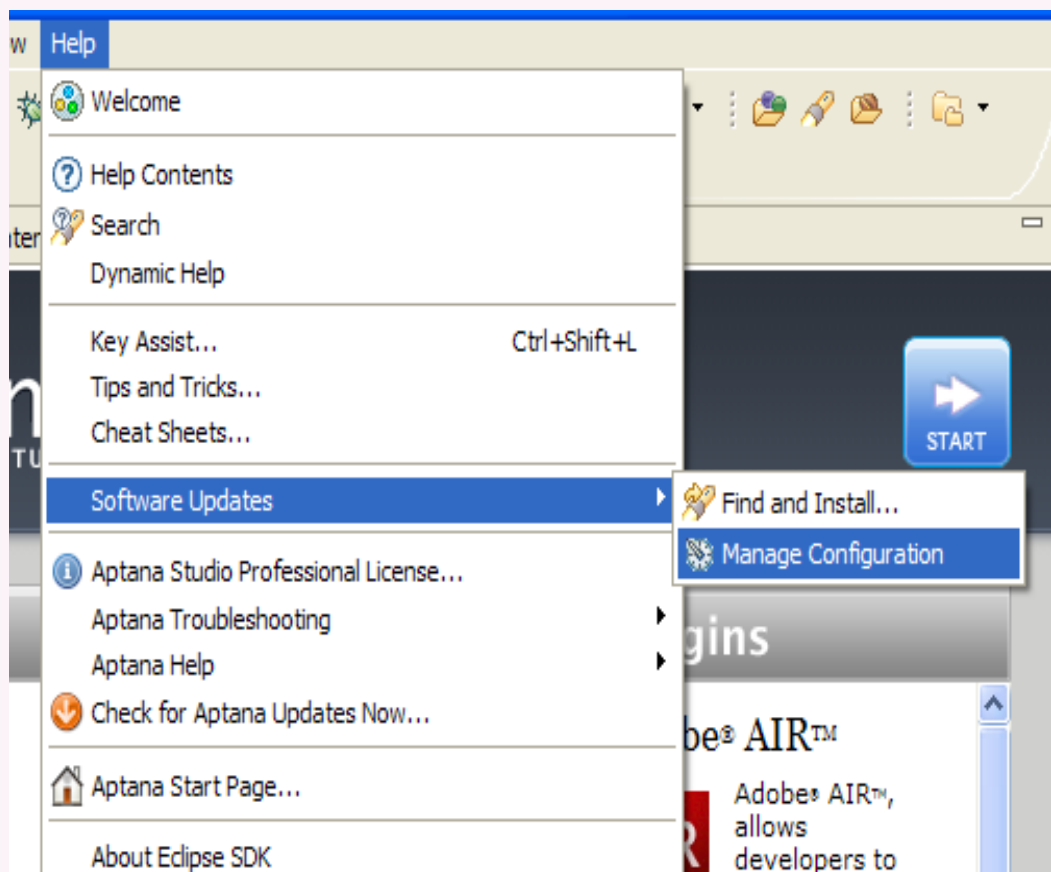


## 4.2.2 Aptana 的安装

需要注意的是：

并不是所有的情况下都能成功安装，很有可能由于 Aptana 的部分插件与已经安装在 Eclipse 里面的插件会冲突。如果发生这种情况，请先把 Eclipse 中有冲突的插件禁用或者卸载掉。

## 4.2.2 Aptana 的安装



选择

【 Help 】菜  
单下的

【 Software  
Updates 】选  
项，如下图 4-  
18 所示：

图 4- 18 查看已安装的插件

## 4.2.2 Aptana 的安装

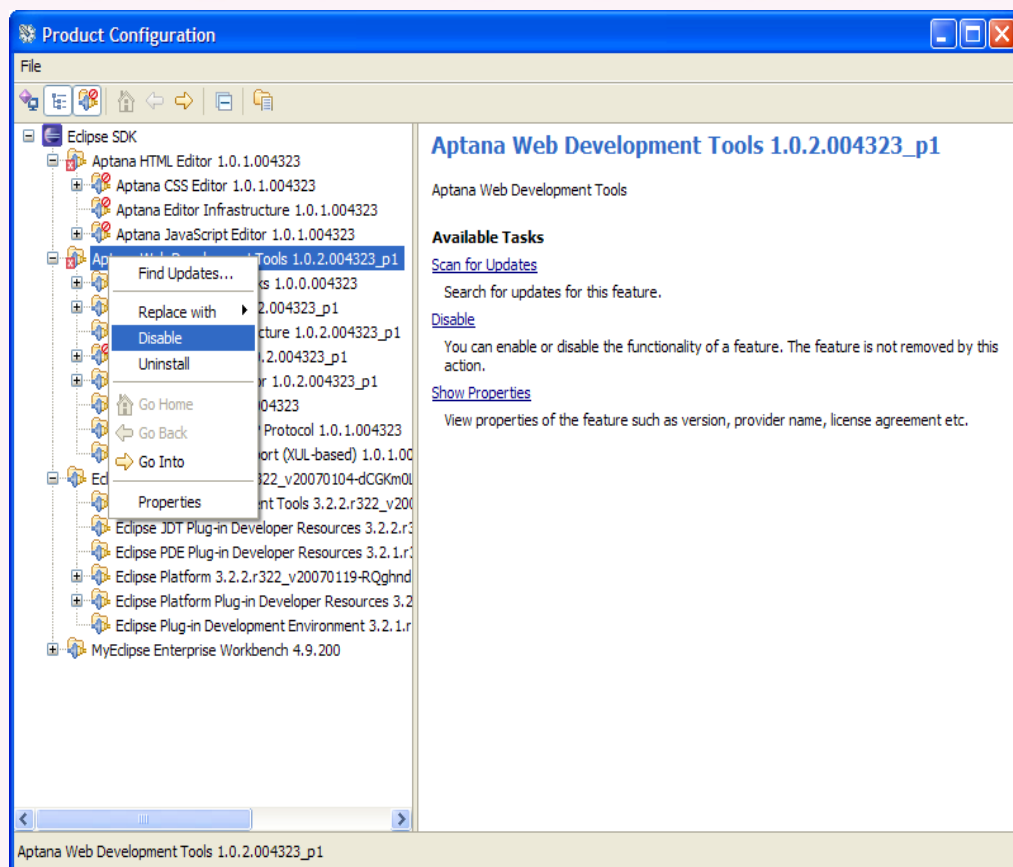


图 4- 19 禁用或者卸载插件

单击【Manage Configuration】菜单项，弹出如图 4- 19 所示对话框，其中列出了所有已经安装的插件，右击当前需要禁用或者卸载的插件，执行相应的操作，然后再重新安装即可。

## 4.2.3 Aptana 使用技巧

### 1. 打开 Aptana 透视

图

如果是安装插件的方法安装的 Aptana，那么默认打开的不是 Aptana 透视图，非 Aptana 透视图的情况，许多 Aptana 的功能都无法直接使用。单击菜单【Window】✉【Open Perspective】✉【Other...】，在弹出的窗口中选择 Aptana，如图 4-20 所示：

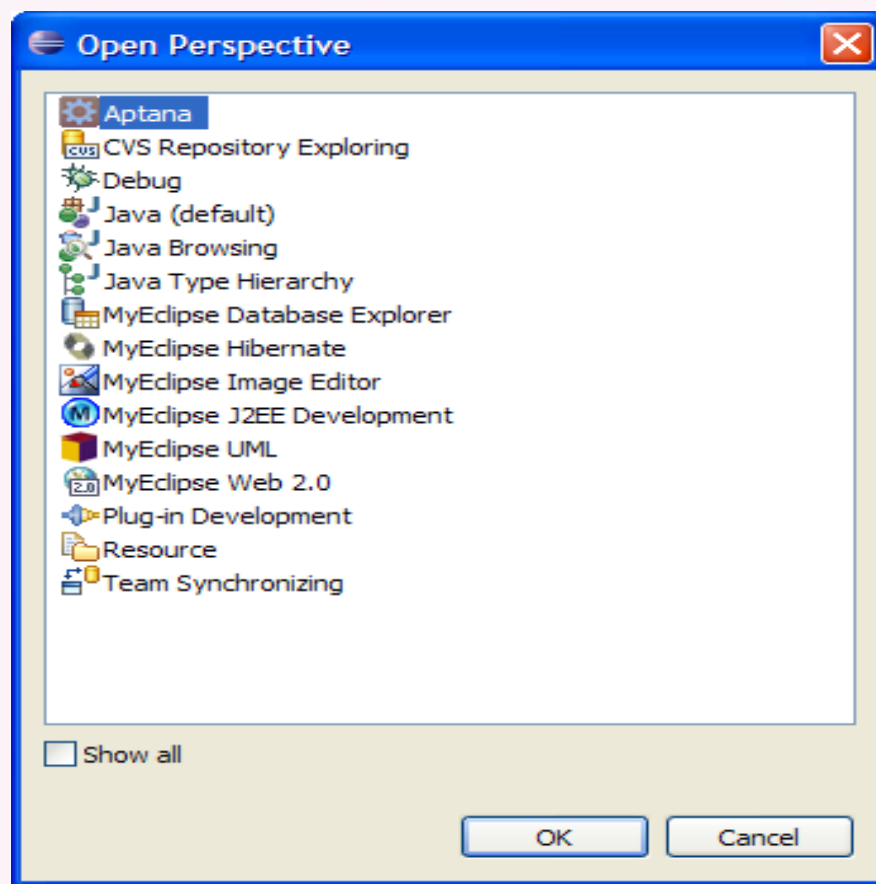


图 4-20 打开 Aptana 窗口透视图

## 4.2.3 Aptana 使用技巧

打开以后，在 Eclipse 的右上角有透视图切换的快捷按钮，如图 4- 21 所示。



图 4- 21 透视图切换窗口

## 4.2.3 Aptana 使用技巧

### 2. 文件关联

默认情况下，Aptana 支持几种常见类型的文件关联打开，如 HTML、JavaScript 等。通过简单的设置，也可以关联其它的文件类型用 Aptana 编辑器打开。

Aptana 的编辑器主要有：Aptana CSS Editor，Aptana HTML Editor，Aptana JS Editor，Aptana Text Editor，Aptana XML Editor。对于网页文件，常用的是 Aptana HTML Editor。

单击菜单中的【Window】☒【Preferences】，在弹出的窗口中，依次单击【General】☒【Editors】☒【File Associations】，进入文件关联设置，如图 4-22 所示：

## 4.2.3 Aptana 使用技巧

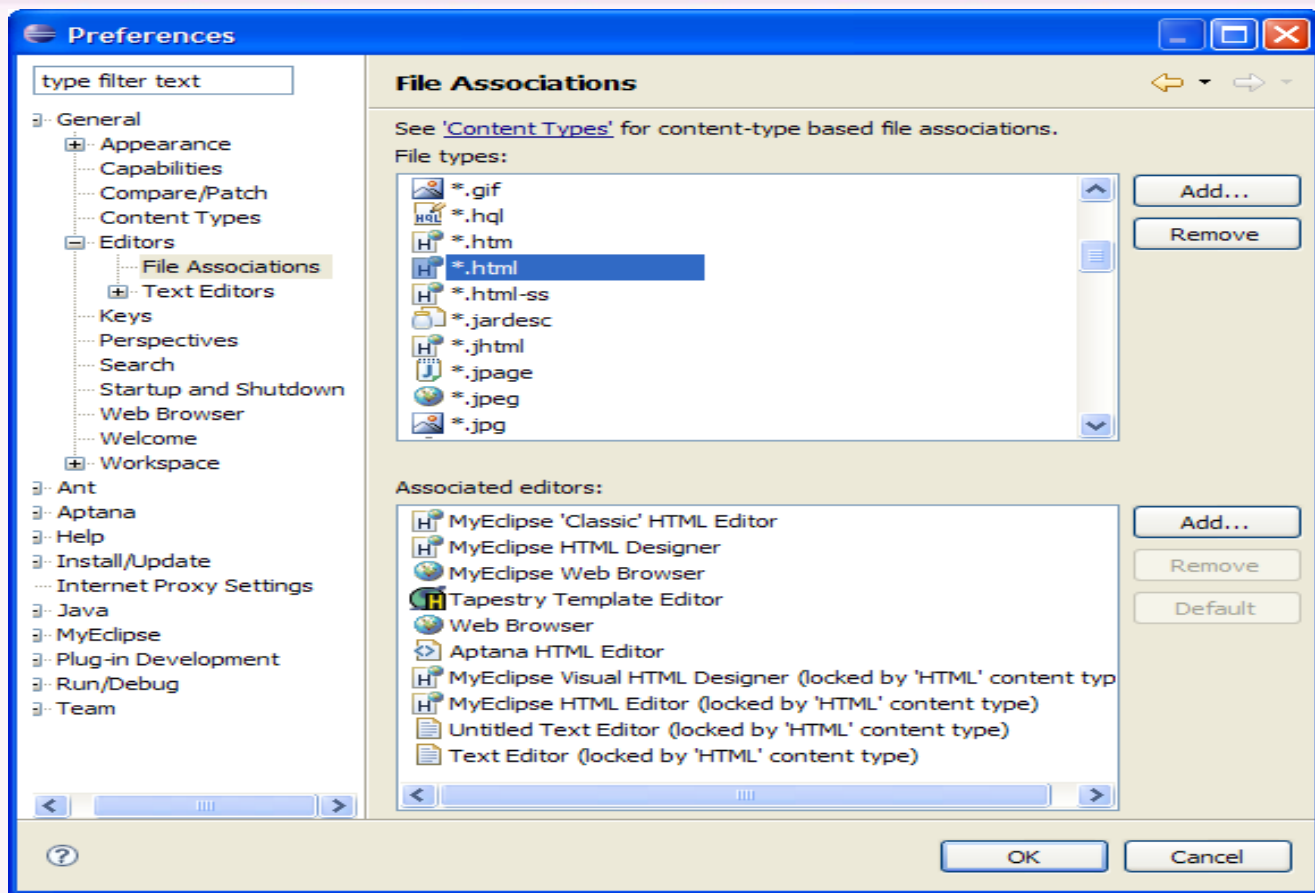


图 4- 22 关联文件编辑器

## 4.2.3 Aptana 使用技巧

在此处进行文件关联的设置，比如设置 html 文件，单击【\*.html】，下方列出关联打开的编辑器，单击 Associated Editors 右边的【Add...】按钮，弹出如图 4- 23 所示的编辑器选择窗口。

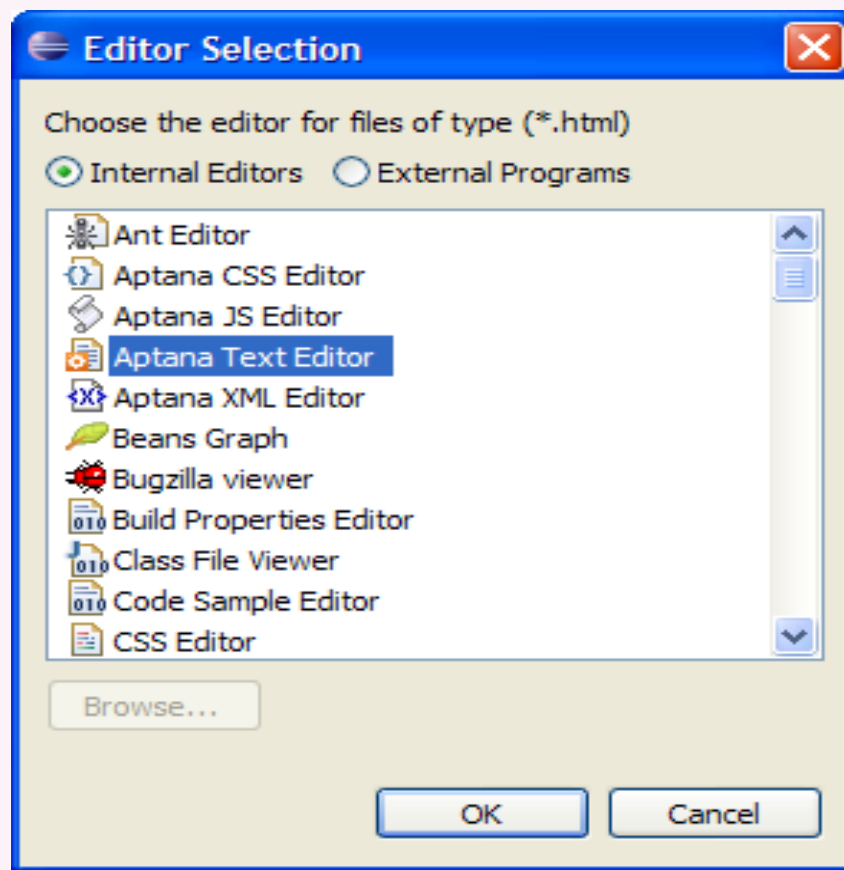


图 4- 23 添加 Aptana Text Editor 编辑器



## 4.2.3 Aptana 使用技巧

在此选择需要关联的编辑器，选后确定，返回文件关联设置界面，如图 4-24 所示：

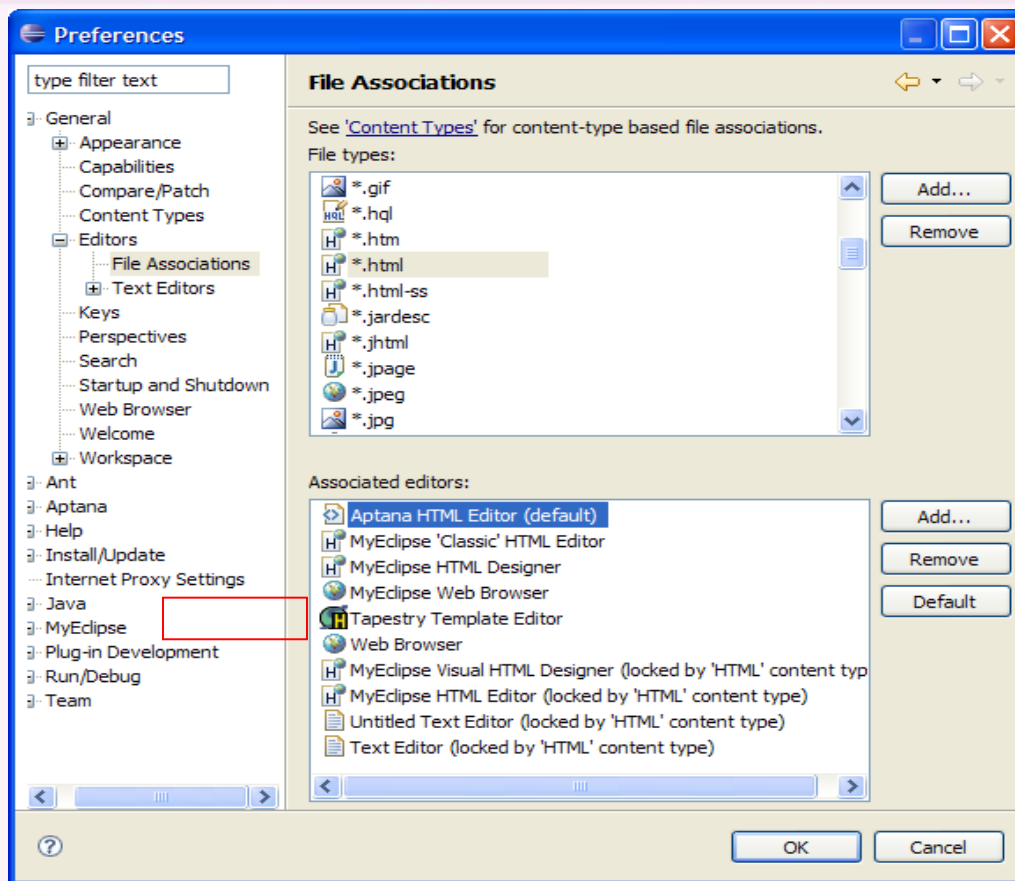


图 4-24 设置默认编辑器

## 4.2.3 Aptana 使用技巧

选中相关编辑器，单击【 default 】按钮，可以把相关的文件用该编辑器默认打开。

右击需要打开的文件，在弹出菜单中单击【 Open With 】，可选择其他编辑器，如图 4-25 所示：

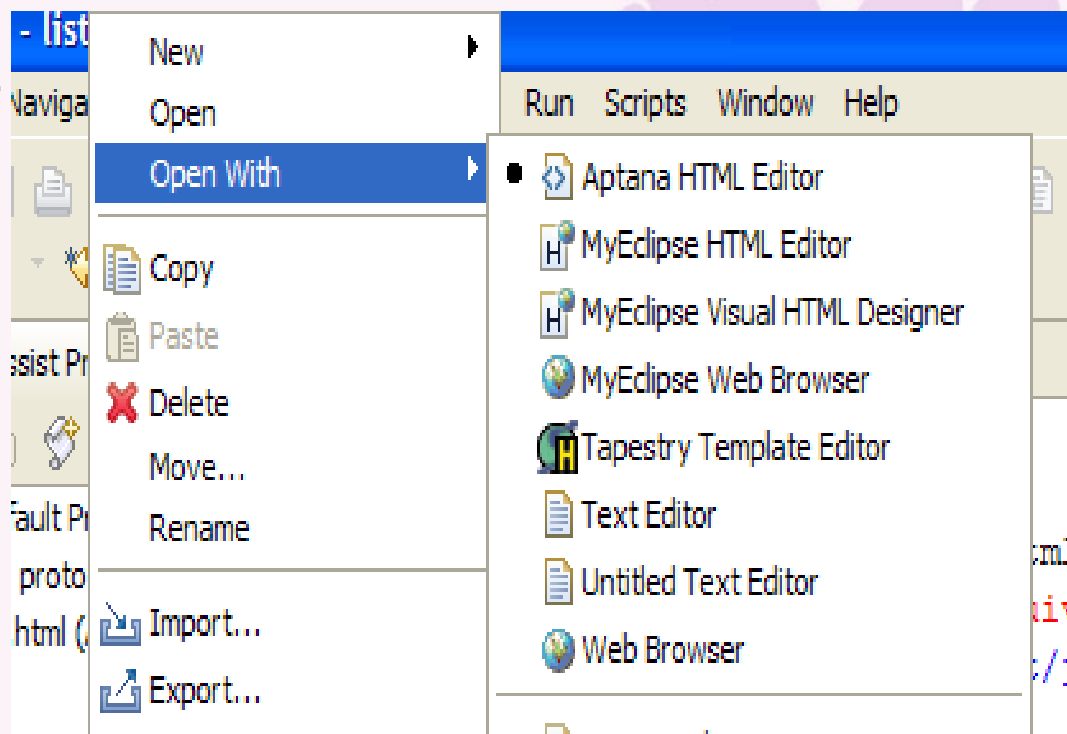


图 4- 25 选择其他编辑器

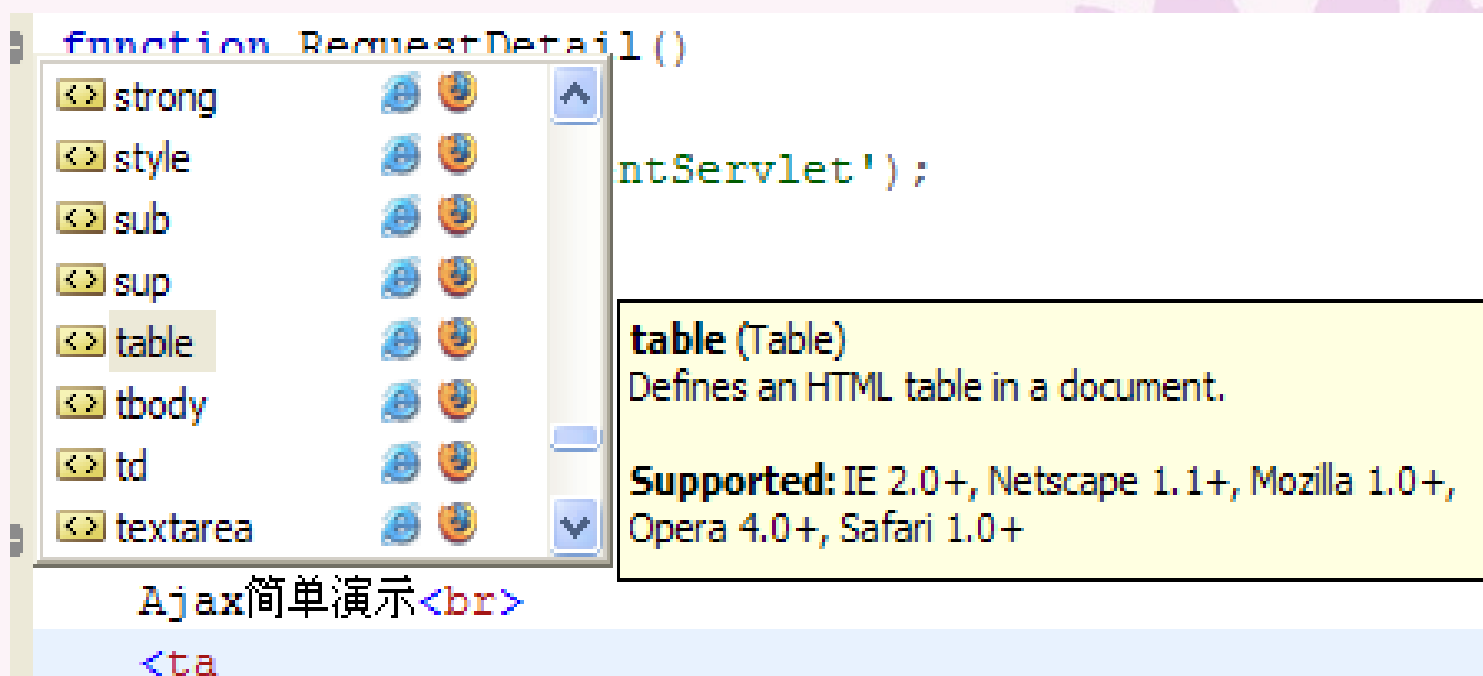
## 4.2.3 Aptana 使用技巧

### 3.Code Assit （ 代码智能提示 ）

Aptana 的代码提示功能非常强大，  
不仅支持标准代码的提示，如  
HTML ， CSS ， JavaScript 的方法和  
属性的提示，而且支持对自定义变量  
和函数的提示。

## 4.2.3 Aptana 使用技巧

( 1 ) 对 HTML 的支持，如图 4- 26 所示：



只要输入“<”等与 HTML 有关的字符，就会自动弹出相关的代码，并且显示相关属性和方法，以及支持的浏览器。

## 4.2.3 Aptana 使用技巧

### ( 2 ) 对 JavaScript 的支持

在 `<script language="JavaScript"></script>` , 输入任何与 JavaScript 代码有关的字符, 就会自动弹出对话框以供选择, 如图 4- 27 所示:

当我们在 document 对象后面输入“get”后, 我们可以发现系统自动提供了一个以 get 开头的函数的下拉对话框以供选择, 我们只需双击某个选项即可选定。

## 4.2.3 Aptana 使用技巧

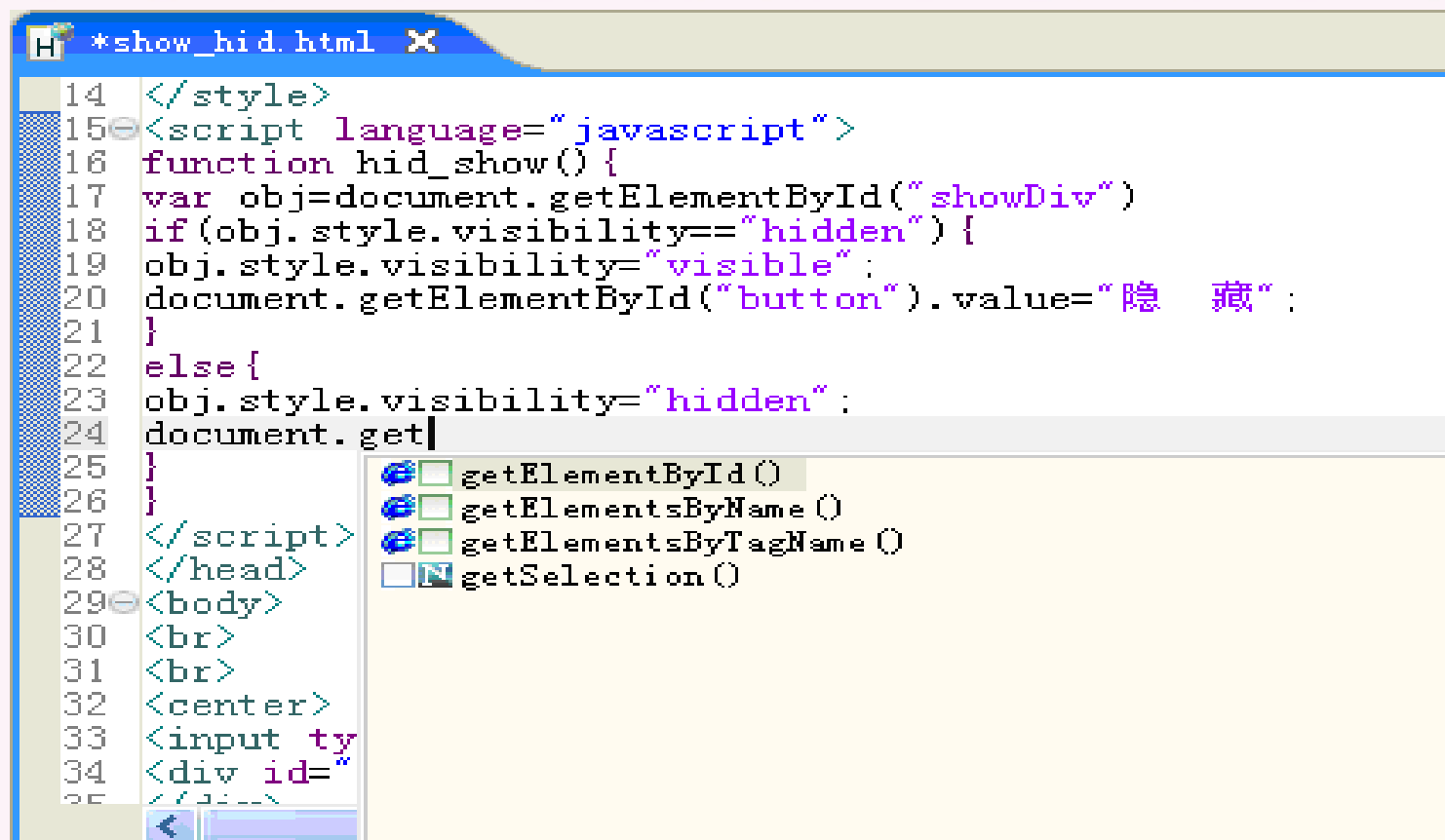


图 4- 27 对 JavaScript 的代码智能提示

## 4.3 JavaScript 的数据类型

JavaScript 中有六种数据类型，分别是数值型、字符串型、布尔型、空值、未定义型、Object。

- ( 1 ) 数值型：JavaScript 支持整数和浮点数。
- ( 2 ) 字符串型：字符串是用单引号或双引号说明的。
- ( 3 ) 布尔型：可能的 Boolean 值有 true 和 false。
- ( 4 ) 空值：null。就是没有任何值，什么也不表示。
- ( 5 ) 未定义型：undefined。
- ( 6 ) Object：对象型。

## 4.3 JavaScript 的数据类型

JavaScript 中变量在声明时并不指定其类型，而是在脚本执行过程中根据需要确定变量的数据类型。

在下面的示例中，最初 myVar 的值是 '变量值是：'，所以 myVar 的数据类型是字符串型；接着又将数值 10 赋值给了 myVar，所以其数据类型变为了数值型。

```
<html>
<head>
<script language="JavaScript">
var myVar='变量值是：';
document.write(myVar);
myVar=10;
document.write(myVar);
</script>
</head>
</html>
```



# 4.4 JavaScript 的运算符

在 JavaScript 中，常用的运算符有以下五种：逻辑运算符、算术运算符、赋值运算符、比较运算符、字符运算符以及条件运算符。

## 1. 逻辑运算符

| 运算符 | 描述 | 例子                                  |
|-----|----|-------------------------------------|
| &&  | 与  | x=6 y=3 (x < 10 && y > 1) 返回值为 true |
|     | 或  | x=6 y=3(x==5    y==5) 返回值为 false    |
| !   | 非  | x=6 y=3 !(x==y) 返回值为 true           |

# 4.4 JavaScript 的运算符

## 2. 算术运算符

| 运算符 | 描述              | 例子                  | 结果          |
|-----|-----------------|---------------------|-------------|
| +   | 加               | x=2<br>y=2<br>x+y   | 4           |
| -   | 减               | x=5<br>y=2<br>x-y   | 3           |
| *   | 乘               | x=5<br>y=4<br>x*y   | 20          |
| /   | 除               | 15/5<br>5/2         | 3<br>2.5    |
| %   | 求余数<br>( 保留整数 ) | 5%2<br>10%8<br>10%2 | 1<br>2<br>0 |
| ++  | 累加              | x=5<br>x++          | x=6         |
| --  | 递减              | x=5<br>x--          | x=4         |

# 4.4 JavaScript 的运算符

## 3. 赋值运算符

| 运算符 | 例子      | 等效写法     |
|-----|---------|----------|
| =   | $x=y$   | $x=y$    |
| +=  | $x+=y$  | $x=x+y$  |
| -=  | $x-=y$  | $x=x-y$  |
| *=  | $x*=y$  | $x=x*y$  |
| /=  | $x/=y$  | $x=x/y$  |
| %=  | $x\%=y$ | $x=x\%y$ |

# 4.4 JavaScript 的运算符

## 4. 比较运算符

| 运算符 | 描述            | 例子                                                |
|-----|---------------|---------------------------------------------------|
| ==  | 等于            | 5==8 返回值为 false                                   |
| === | 等于 ( 检查值和类型 ) | x=5 y="5"<br>x==y 返回值为 true<br>而 x===y 返回值为 false |
| !=  | 不等于           | 5!=8 返回值为 true                                    |
| >   | 大于            | 5>8 返回值为 false                                    |
| <   | 小于            | 5<8 返回值为 true                                     |
| >=  | 大于等于          | 5>=8 返回值为 false                                   |
| <=  | 小于等于          | 5<=8 返回值为 true                                    |

## 4.4 JavaScript 的运算符

### 5. 字符运算符

字符串通常是指文本，比如 "Hello World!"。使用运算符 + 就可以将两个字符变量结合在一起：

```
var a="She is";  
var b="a pretty girl!";  
var c=a+b;  
document.writeln(c);
```

## 4.4 JavaScript 的运算符

### 6. 条件运算符

JavaScript 也包含条件运算符，这种运算符可以给基于条件的变量赋值。

语法为：变量名 =( 条件 )? 值 1: 值 2

```
result=(average>=60)?"easy":"difficult";
```

如果变量 average 大于等于 60，那么给 result 赋的值是“easy”；假如变量 average 小于 60，那么给 result 赋的值是“difficult”。

# 4.5 JavaScript 的对象

## 4.5.1 什么是JavaScript 中的对象

- JavaScript 是基于 Java 程序设计语言而建立起来的脚本语言，它是基于对象的编程，但还不是完全的面向对象的编程。
- JavaScript 中的对象是由属性和方法两个基本的元素所构成的。
- JavaScript 中使用 new 关键字来实例化一个类，生成一个对象。生成对象实例的格式和写法如下：

```
var str = new String("This is String."); // 生成一个字符串对象
```

```
var now = new Date(); // 生成一个日期对象
```

- 对象的属性可以通过 < 对象名 >.< 属性 > 来访问。如：  
var str = new String("This is String."); // 新建一个  
//String 对象  
var len = str.length; //length 为字符串长度属性
- 同样，对象的方法也可以通过 < 对象名 >.< 方法名 >() 来访问，不过在方法名后面要有一对圆括号，括号里面应该包含了使用该方法所需要传递的参数。

如：

```
var str1 = new String("This is str1.");  
var str2 = new String("This is str2");  
var str3 = str1.concat(str2); //concat 方法用于
```

// 字符串连接，返回字符串连接之后的

值

变量 str3 的值将是 "This is str1.This is str2"。



## 4.5.2 JavaScript 内置对象

JavaScript 本身提供了一些内置的对象和函数，内置对象提供编程最常用的功能。常用的 JavaScript 内置对象有以下几种：

Date 对象：处理日期和时间的存储、转化和表达。

String 对象：处理所有的字符串操作。

Array 对象：提供一个数组模型、存储大量有序的数据。

## 4.5.2 JavaScript 内置对象

内置对象都有自己的方法和属性，访问的方法如下：

对象名. 属性名称

对象名. 方法名称 ( 参数列表 )

## 4.5.2 JavaScript 内置对象

### (1)Date（日期和时间）对象

必须使用 new 运算符创建一个 Date 对象实例，才能使用。  
Date 对象的使用方法如下：

```
var myDate = new Date();  
// 定义 myDate 为日期对象，且已有初始值：当前时间
```

如果要自定初始值，可以用：

```
var d = new Date(99, 10, 1);    //99 年 10 月 1 日  
var d = new Date('Oct 1, 1999'); //99 年 10 月 1 日
```

## 示例：状态栏上显示时间

```
<html>
<head>
<title> 状态栏上显示时间 </title>
<script language="JavaScript">
var timerID = null;
var timerRunning = false;
function stopclock (){
    if(timerRunning)
        clearTimeout(timerID);
    timerRunning = false;
}
function showtime () {
    var now = new Date();
    var hours = now.getHours();
    var minutes = now.getMinutes();
    var seconds = now.getSeconds();
```

```
var timeValue = "" + ((hours > 12) ? hours - 12 : hours)
timeValue += ((minutes < 10) ? ":0" : ":") + minutes
timeValue += ((seconds < 10) ? ":0" : ":") + seconds
timeValue += (hours >= 12) ? " P.M." : " A.M."
window.status = timeValue;
timerID = setTimeout("showtime()", 1000);
timerRunning = true;
```

```
}
function startclock () {
    stopclock();
    showtime();
}
```

```
</script>
```

```
</head>
```

```
<body onLoad="startclock()">
```

```
<p> 显示当前系统时间代码示例  </p>
```

```
</body>
```

```
</html>
```

## (2) String ( 字符串 ) 对象

字符串对象具有很多函数，可以用来处理字符串。字符串对象的建立可以通过两种方式，一是直接赋值，二是通过 new 运算符来创建。格式如下：

格式 1：字符串对象名称 = new String( 字符串常量 )

格式 2：字符串变量名称 = " 字符串常量 "

## 示例：状态栏走马关灯式效果

```
<html><head><title> 状态栏上走马灯效果 </title>
  <SCRIPT Language="JavaScript">
    var msg=" 欢迎使用本书 ";
    var interval = 300;seq = 0;
    function Scroll()
      {len = msg.length;window.status = msg.substring(0,
seq+1);
      seq++;
      if ( seq >= len ) { seq = 0 };
      window.setTimeout("Scroll();", interval );}
  </SCRIPT>
</head>
<body onLoad="Scroll()">
</body>
</html>
```

### ( 3 ) Array ( 数组 ) 对象

- 数组对象是一个对象的集合，里边的对象可以是不同类型的。数组的每一个成员对象都有一个“下标”，用来表示它在数组中的位置，下标从零开始。

- 数组的创建格式如下：

```
var arrayObj = new Array()
```

```
var arrayObj = new Array([ 大小 ])
```

```
var arrayObj = new Array([ 元素 1[, 元素 2[, ...[, 元素  
N]]])
```

- 数组创建之后，能够用 [] 符号访问数组单个元素，例：

```
var myArray = new Array();for (i = 0; i < 10; i++){  
myArray[i] = i;}var x = myArray[4];
```



## 数组对象使用示例：动态导航菜单

```
<html>
<head>
<title> 动态导航菜单 </title>
</head>
<body>
<script language='JavaScript'>
var index = 7;
link = new Array(6);
text = new Array(6);
link[0] ='sample.htm';
link[1] ='sample.htm';
link[2] ='sample.htm';
link[3] ='sample.htm';
link[4] ='sample.htm';
link[5] ='sample.htm';
link[6] ='sample.htm';
```

```
text[0] = ' 菜单一 ';
text[1] = ' 菜单二 ';
text[2] = ' 菜单三 ';
text[3] = ' 菜单四 ';
text[4] = ' 菜单五 ';
text[5] = ' 菜单六 ';
text[6] = ' 菜单七 ';
document.write("<marquee scrollamount='1' scrolledelay='100'
direction= 'up' width='150' height='150'>");
for (i=0;i<index;i++){
document.write ("&nbsp;<a href="+link[i]+" target='_blank'>");
document.write (text[i] + "</A><br>");
}
document.write("</marquee>");
</script>
</body>
</html>
```

## 4.5.3 JavaScript 浏览器对象

JavaScript 最强大的功能也就在于能够直接访问浏览器窗口对象及其子对象。在 JavaScript 能够涉及的范围内有如下几个大的对象：window、document、location、navigator、history 等。

### 1. window（窗口）对象

window 窗口对象是最大的对象，它描述的的是一个浏览器窗口。由于窗口对象是最高层对象，因此一般要引用它的属性和方法时，可以不必假如对象名 window。

## ( 1 ) 窗口的常用属性

窗口的常用属性包括

name、 status、 opener、 parent、 top。

- status : 指窗口下方的“状态栏”所显示的内容。通过对 status 赋值, 可以改变状态栏的显示信息。我们常常见到的状态栏走马灯式的效果就是由此实现的。
- opener : 返回打开本窗口的窗口对象。如果窗口不是由其他窗口打开的, 在 Netscape 中这个属性返回 null ; 在 IE 中返回“未定义”( undefined )。
- parent : 返回窗口所属的框架页对象。
- top : 返回占据整个浏览器窗口的最顶端的框架页对象。

表 4-6 窗口属性参数

| 参数          | 含义                   |
|-------------|----------------------|
| top         | 窗口顶部离开屏幕顶部的像素数       |
| left        | 窗口左端离开屏幕左端的像素数       |
| width       | 窗口的宽度                |
| height      | 窗口的高度                |
| menubar     | 窗口有没有菜单，取值 yes 或 no  |
| toolbar     | 窗口有没有工具条，取值 yes 或 no |
| location    | 窗口有没有地址栏，取值 yes 或 no |
| directories | 窗口有没有连接区，取值 yes 或 no |
| scrollbars  | 窗口有没有滚动条，取值 yes 或 no |
| status      | 窗口有没有状态栏，取值 yes 或 no |
| resizable   | 窗口可否调整大小，取值 yes 或 no |

## ( 2 ) 窗口的常用方法

- `open(<URL>, <窗口名称>, <窗口属性参数>)` : 打开一个窗口。
- `<URL>` : 描述打开的窗口所访问的网页地址。如果留空 ( " ) , 则不打开任意网页。
- `<窗口名称>` : 描述被打开的窗口的名称。
- `<窗口属性参数>` : 描述被打开的窗口的样式。如果只需要打开一个普通窗口, 该字符串留空 ( " ) , 如果要指定样式, 就在字符串里写上一到多个参数, 参数之间用逗号隔开。

( 3 ) 示例 :

打开一个新窗口

```
Var mywin=window.open(", '_blank', 'width=200,  
height=200,menubar=no,toolbar=no,location=no,dire  
ctories=no,status=no,scrollbars=yes,resizable=yes');
```

关闭一个窗口

```
mywin.close();
```

弹出一个只包含“确定”按钮的对话框，显示 < 字符串 > 的内容

```
window.alert('this is a alert test');
```

弹出一个包含“确定”和“取消”按钮的确认对话框。如果用户按下“确定”，则返回 true 值，如按下“取消”，则返回 false 值，可以根据返回值来进行不同的操作。

```
window.confirm(' 您是否要删除本条记录？ ');
```

弹出一个包含“确认”、“取消”和一个文本框的对话框，显示 < 提示信息 > 的内容，要求用户在文本框输入一些数据。如果用户按下“确认”，则返回文本框里已有的内容，如果用户按下“取消”，则返回 null 值。如果指定 < 初始值 >，则文本框里会有默认值

```
var name=window.prompt('please enter your name');
```



## 2. history（历史）对象

历史对象描述了浏览器的浏览历史，通过历史对象可以使浏览器跳转到曾经访问过的站点或页面。

### （1）历史对象的常用属性

next：下一个文档。

previous：前一个文档。

### （2）历史对象的常用方法

back()：与“后退”按钮等效。

forward()：与“前进”按钮等效。

go(val)：在历史的范围内去到指定的一个地址。如果  $val < 0$ ，则后退  $val$  数量的地址，如果  $val > 0$ ，则前进  $val$  数量的地址，如果  $val = 0$ ，则刷新现在打开的网页。

### 3. location （地址）对象

地址对象描述的是某一个窗口对象所打开的地址，可以直接使用“location”来表示当前打开的窗口地址。

#### （1）地址对象的常用属性

href：返回整个地址。也就是在浏览器的地址栏上显示的内容。我们常常可以使用 href 来进行页面跳转，使用格式如下：

```
document.location.href = 'http://www.163.com';
```

host : 返回主机名和端口号。

hostname : 返回地址的主机名。例如，一个“http://www.163.com/blog/”的地址，location.hostname = 'www.163.com'。

port : 返回地址的端口号，一般http的端口号是“80”。

## ( 2 ) 地址对象的常用方法

reload() : 重新加载页面内容，history.go(0) 是等效的。

## 4. document（文档）对象

文档对象描述当前窗口或指定窗口对象的文档。它包含了文档从 `<head>` 到 `</body>` 的内容。利用它可以控制文档中的 HTML 标识，它本身还包含了许多子对象。

### （1）文档对象的常用属性

`forms[]`：返回文档中所有表单对象所组成的数组。每一个元素对应于文档中的一个 `<form>` 标签，其访问格式如下：

```
var myVar = document.forms[0];
```

由于在 JavaScript 中数组的下标也是从 0 开始的，所以变量 myVar 指向网页文档中的第 1 个 <form> 标签对应的表单对象。JavaScript 还提供另外的方式，通过表单的名字来访问，前提是为该表单对象加一个 name 的属性。

```
<form name="myForm"></form>
```

以下方式便可以访问名为“myForm”的表单对象了。

```
var myVar = document.myForm;
```

Tips: 在使用以上属性访问表单对象的时候，要注意不要有重名的情况

其它常用属性还有：

images[]、 links[]、 bgcolor、 fgcolor、 last modified、 linkcolor

## ( 2 ) 文档对象的常用方法

- getElementById ( id ) : 该方法可返回对拥有指定 ID 的第一个对象的引用, 其他类似的还有 : getElementsByName ( name ) getElementsByTagName ( tagname ) 等。
- write(<string1[,string2,...]>) : 向文档中写入数据。该字符串可以是普通文本, 也可以是 HTML 标识。
- writeln(<string1[,string2,...]>) : 同 write() 方法一样, 不同的只是 writeln() 方法会在文本之后假如换行符。
- clear() : 清空当前文档, 也将清除脚本。

## 5. form（表单）对象

form 对象用于存放网页中的 document 对象的子对象。单域元素，如文本框、文本、复选框、按钮等。JavaScript 能就是能直接处理表单中的，可以通过下面一条语句来显示的内容：

Tips: 表单对象的常用方法只有一个 submit()，该方法的主要功用是实现表单信息的提交

```
document.myForm.username.value='hello';  
// username 是文本框的名称。
```

## 4.6 JavaScript 的函数

在 JavaScript 中支持两类函数，一类是语言内部的函数，另一类是自己创建的。如果需要创建自己的函数，可以使用 function 语句。格式如下：

```
function 函数名 ([ 参数 [, 参数..... ]]) {  
    语句组  
    [return < 表达式 >;]  
}
```

在函数中可以使用“return 表达式”语句来返回一些值。函数名必须复合变量名的命名规则，最好能易于识别。函数的位置应该按照逻辑顺序，集中置顶，处于（<head>...</head>）之间。函数在调用的时候，使用函数名以及传入函数所需要的参数即可。



示例：

```
<script language="JavaScript">
function showName(name){
    document.write(name );
    alert(" 你的名字是 " + name);
}
showName(" 张三 ");
</script>
```

我们在使用函数的时候，需要注意以下几点：

- 函数由关键字 `function` 定义；
- 函数必须先定义后使用，否则将出错；
- 函数名是调用函数时引用的名称，它对大小写是敏感的，调用函数时不可写错函数名；
- 参数表示传递给函数使用或操作的值，它可以是常量，也可以是变量；
- `return` 语句用于返回表达式的值，也可以没有。
- 此外，JavaScript 本身也提供的大量的函数（对象的方法），能够实现很多的功能，在下一节内容中会做介绍。

# 4.7 JavaScript 的流程控制

## 4.7.1 条件语句

### if 语句

JavaScript 支持 if 和 if...else 条件语句。在 if 语句中将测试一个条件，如果该条件满足测试，执行相关的语句组。在 if...else 语句中，如果条件不满足测试，则将执行不同的代码。

格式：

```
if (< 条件 1>){  
    < 语句组 1>  
}  
[else if (< 条件 2>){  
    < 语句组 2>  
}]  
[else{  
    < 语句组 3>  
}]
```

示例：通过输入的分数来查看成绩等级。

```
<html><head>
<title> 通过输入的分数来查看成绩等级 </title>
<script language="JavaScript">
function setGrade()
{
    var score = document.form1.score.value;
    var result="";
    if (score >= 0 && score < 60) {
        result = 'fail';
    } else if (score < 80) {
        result = 'pass';
    } else if (score < 90) {
        result = 'good';
    }
}
```

```
} else if (score <= 100) {  
    result = 'excellent';  
} else {  
    result = 'error!';  
}  
    document.form1.grade.value = result;  
}  
</script>  
</head><body>  
<form name="form1">  
<input type="text" name="score" value="">  
<input type="button" value=" 查看等级 "  
onclick="setGrade()"/>  
<input type="text" name="grade" value="">  
</form></body></html>
```

## 2 条件运算

JavaScript 也支持条件运算。条件运算在要测试的条件后使用一个问号，在问号后面指定两个可选项，第一个在满足条件时使用，第二个在条件不满足时使用。这两个选择项之间用一个冒号隔开。

格式：

( 条件 ) ?< 选项 1>:< 选项 2>

示例：

```
var myVar = "";  
myVar = (X == "A") ? "B" : "C";  
// 如果 X 的值为 A ， myVar 就是 B ，否则 myVar 就是 C。
```

## 3 switch 语句

格式：

```
switch (< 表达式 >) {  
    case < lable >:< 语句组 1>  
        break;  
    case < lable >:< 语句组 2>  
        break;  
    ...  
    default: < 语句组 >  
}
```

其中 label 是根据 < 表达式 > 来匹配的标识符。  
通过 switch 语句执行流程如下：

- ① 求 < 表达式 > 的值并依次序查看 label，直到找到一个匹配。
- ② 如果 label 的值等于 < 表达式 > 的值，则执行它相应的 < 语句组 >。
- ③ 继续执行，直到遇到一个 break 语句，或者 switch 语句结束。这意味着如果没有使用一个 break 语句，则多个 label 块被执行。
- ④ 如果没有 label 等于 < 表达式 > 的值，则跳转到 default 情况。如果没有 default 情况，则跳转到最后一步。
- ⑤ 继续执行紧接 switch 代码块末尾的语句。



上面的示例查看成绩等级情况可以改为如下代码：

```
<html><head>
<title> 通过输入的分数查看成绩等级情 </title>
<script language="JavaScript">
function setGrade()
{
    var score = document.form1. score.value;
    var result="";
    switch (parseInt(score / 10))
    {
        case 0:
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:
            result = 'fail';
            break;
```

```
case 6:  
case 7:  
    result = 'pass';  
    break;  
case 8:  
    result = 'good';  
    break;  
case 9:  
    result = 'excellent';  
    break;  
default:  
    if (score == 100)  
        result = 'excellent';  
    else  
        result = 'error!';  
}
```

```
        document.form1.grade.value = result;
    }
</script>
</head>
<body>
<form>
<input type="text" name="score" value="">
<input type="button" value=" 查看等级 "
onclick="setGrade()"/>
<input type="text" name="grade" value="">
</form>
</body>
</html>
```

## 4.7.2 循环语句

### 1. while 循环

while (< 表达式 >)

{

语句组

}

示例：

```
<script>
var i=0;
while(i<=10){
  document.write("i= "
,i,"<BR>");
  i++;
}
</script>
```

## 2. do...while 循环

用法与 while 循环类似，只是 do...while 循环至少会执行一次。 格式：

do

{ 语句组 }

while (< 表达式 >);

```
<script>
var i=0;
do {
    document.write("i= " ,i ,"<BR>");
    i++;
} while(i<=10)
</script>
```

### 3. for 循环

for 语句指定了一个计数器变量，一个测试条件，以及更新该计数器的操作。格式：

```
for([ 初始表达式 ];[ 条件表达式 ];[ 更新表达式 ])  
{ 语句组 }
```

示例：

```
<script>  
for(var i= 0;i<=10;i++) {  
    document.write("i= " ,i,"<BR>");  
}  
</script>
```

## 4. for...in 循环

for...in 可以遍历一个对象的所有用户定义的属性或者一个数组的所有元素。该方式中的循环计数器是一个字符串，而不是数字。格式：

```
for ( 变量 in 数组 / 对象 ){  
    < 语句组 >}
```

示例：

```
<script>  
var myArray=["one","two","three"];  
for ( str in myArray) {  
    document.writeln(myArray[myVar] ,"<BR>");  
}  
</script>
```

# 4.8 JavaScript 事件驱动

## 4.8.1 基本概念

JavaScript 是基于对象（ object-based ）的语言。这与 Java 不同，Java 是面向对象的语言。而基于对象的基本特征，就是采用事件驱动（ event-driven ）。它是在有形界面的环境下，使得一切输入变化简单化。通常鼠标或热键的动作我们称之为事件（ Event ），而由鼠标或热键引发的一连串程序的动作，称之为事件驱动（ Event Driver ）。而对事件进行处理程序或函数，我们称之为事件处理程序（ Event Handler ）。在 JavaScript 中对象事件的处理通常由函数（ function ）担任。



## 4.8.2 事件驱动

JavaScript 事件驱动中的事件是通过鼠标或热键的动作引发的。它主要有以下几个事件：

### 1. 单击事件 onClick

当用户单击鼠标按钮时，引发 onClick 事件，同时 onClick 指定的事件处理程序或代码将被调用执行。onClick 事件通常在下列控件对象中产生：

checkbox、radio、reset、submit、button 等。

例如：可通过 button 按钮激活 check()，以下是引用片

```
<input type="button" Value=" 检查" onClick="check()">
```

在 onClick 的等号后，可以使用自己编写的函数作为事件处理程序，也可以使用 JavaScript 中内部的函数，也可以直接使用 HTML 中已有的代码等。例如：

```
<input type="button" value=" 提示 " onclick="alert(' 这是一个例子 ')" >
```

## 2. 改变事件 onChange

当利用 text 或 textarea 等控件的输入值改变时引发该事件，或者当 select 选择项中的选中状态改变后也会引发该事件。例：

```
<input type="text" name="Test" value="Test"  
onChange="check()">
```

## 3. 提交事件 onSubmit

当表单中的确认（ submit ）按钮被点击时，引发改事

```
<form method="post" action="reg" name="form1"  
onSubmit="return check()">  
<input name="B1" type="submit" value=" 提交 ">  
</form>
```

## 4. 选中事件 onSelect

当 Text 或 Textarea 对象中的文字被加亮后，引发该事件。

```
<input type="text" value="Hello world!"  
onSelect="alert(' 您已经选择了文本框的内容 ')" />
```

Tips:HTML 标记的属性值要用一对双引号括起来，但有时因为没有正确设定双引号的匹配，导致程序出错，如上面例子中的 onselect="alert('.....')" 如果写成 onselect="alert(".....")" 就是错误的。浏览器解析这段代码时，会认为 "alert(" 是 onselect 的属性值，后面的内容都被“截断”。

## 5. 获得焦点事件 onFocus

当前 text、textarea 等对象获得焦点时引发该事件，常见的例子如光标落在某个 text 输入框里等

```
<html>
<head>
<script language="JavaScript">
function setStyle(x){
document.getElementById(x).value=" 请输入用户名 "
}
</script>
</head>
<body>
<input type="text" onFocus="setStyle(this.id)" id="username"
size="30" />
</body>
</html>
```

## 6. 失去焦点 onBlur

当 text、textarea 等对象不再拥有焦点时引发该事件，与 onFocus 事件是相反的。

## 7. 载入文件 onLoad

当文档载入时，产生该事件。onLoad 一个作用就是在首次载入一个文档时检测 cookie 的值，并用一个变量为其赋值，使它可以被源代码使用。

```
<html>
<head>
<script language="JavaScript">
function load()
{
window.status="Page is loaded"
}
</script>
</head>
<body onload="load()">
</body>
```

## 8. 卸载文件 onUnload

当 Web 页面退出时引发 onUnload 事件，与 onLoad 事件相反。

## 9. 按下键盘按键 onKeyDown

当用户按下键盘按键时引发 onKeyDown 事件。

```
<form>  
<input type="text" onkeydown="check()" />  
</form>
```

其他的还有

onKeyPress、onKeyUp、onMouseMove、onMouseOut、onMouseOver、onMouseUp、onMouseDown、onDbClick 等

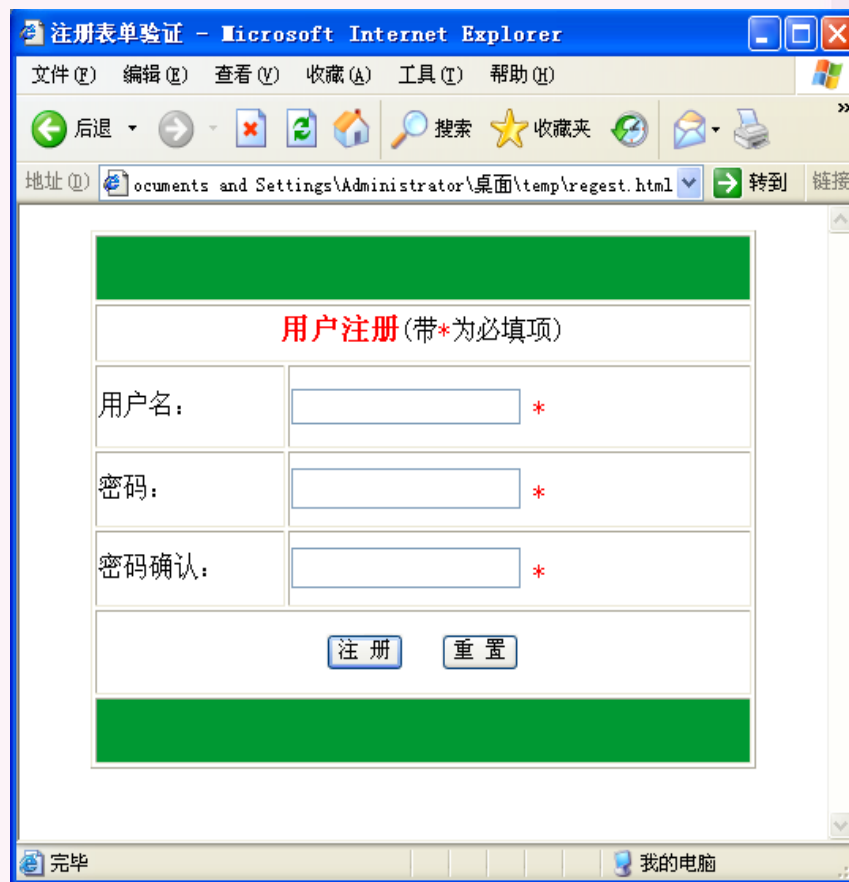


# 4.9 JavaScript 事件处理

## 4.9.1 注册表单验证的实

### 例

在论坛系统的用户注册功能中，如果用户忘记填写必填信息，如用户名、密码等，浏览器会弹出警告框，提示用户当前有未填信息。这个典型的应用就是通过JavaScript实现的。如图4-28所示是一个简单的用户注册页面：



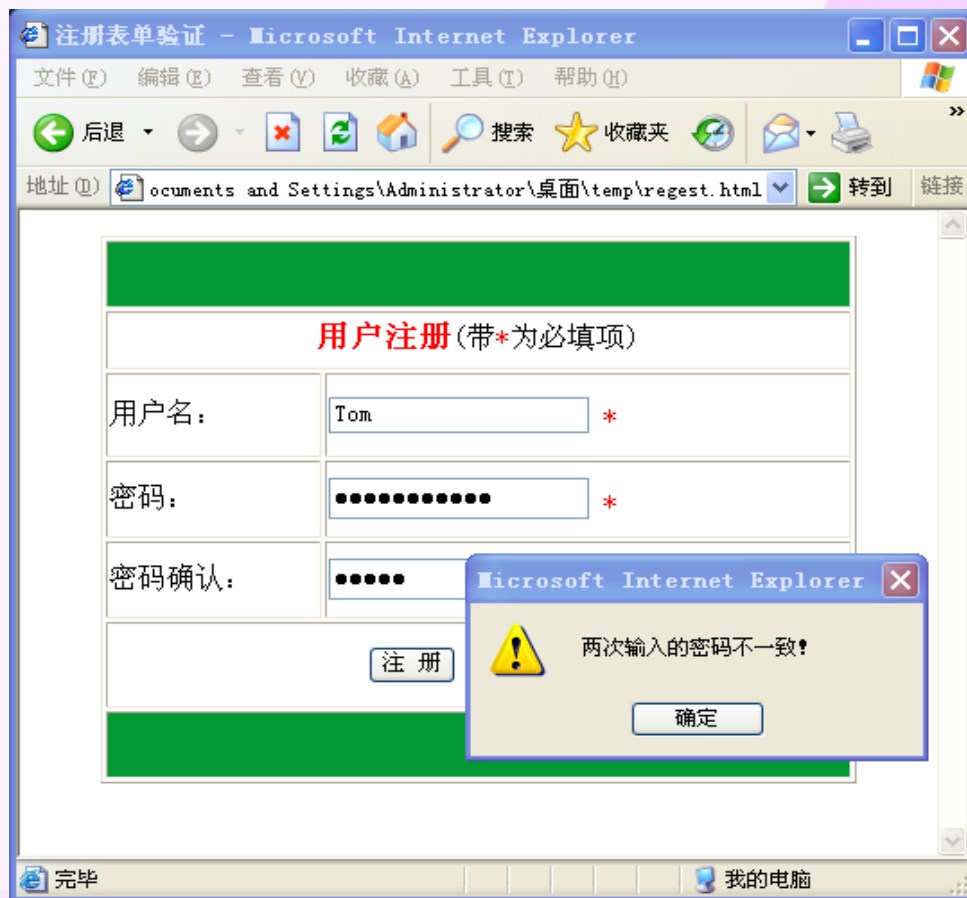
The screenshot shows a Microsoft Internet Explorer window titled "注册表单验证 - Microsoft Internet Explorer". The address bar displays "documents and Settings\Administrator\桌面\temp\regist.html". The page content includes a green header bar, a title "用户注册 (带\*为必填项)", and three input fields labeled "用户名:", "密码:", and "密码确认:". Each input field has a red asterisk (\*) to its right, indicating it is a required field. Below the input fields are two buttons: "注册" (Register) and "重置" (Reset). The browser's status bar at the bottom shows "完毕" (Finished) and "我的电脑" (My Computer).

按要求用户必须输入用户名、密码和确认密码，而且两次输入的密码必须相同，否则，系统会提示错误。当用户没有输入用户名就提交注册时，系统会弹出如图 4- 29 所示的警告窗口，强制用户必须输入



同样，当用户没有输入密码就提交注册时，系统会弹出如图 4- 30 所示的警告窗口，强制用户必须输入密码

如果用户没有输入确认密码或确认密码和前一次输入的密码不同就提交注册，系统就会弹出如图 4- 31 所示的警告窗口，强制用户必须输入正确的确认密码（代码见下一页）：



```
<html><head>
<meta http-equiv="Content-Type" content="text/html;
charset=gb2312" />
<title> 注册表单验证 </title>
<script language="JavaScript">
function check(){
if(document.form1.username.value == ""){
alert(" 用户名不能为空！ ");
document.form1.username.focus();
return false;
}
if(document.form1.password.value == ""){
alert(" 密码不能为空！ ");
document.form1.password.focus();
return false;}
}
```

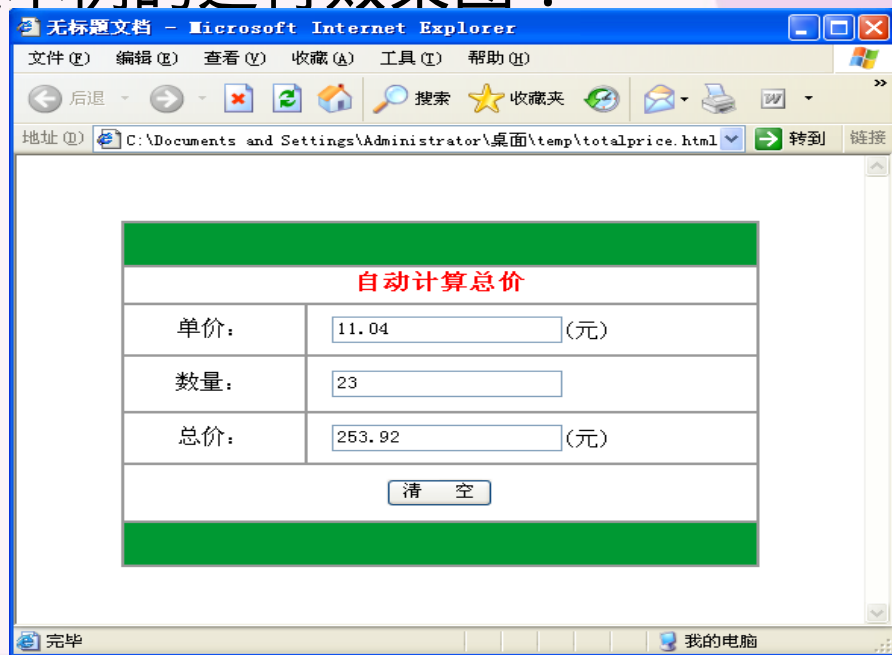
```
if(document.form1.repassword.value!=
document.form1.password.value){
    alert(" 两次输入的密码不一致！ ");
    document.form1.repassword.focus();
    return false;
}
</script></head>
<body>
<form action="" name="form1" onsubmit="return
check()">
    <table width="421" border="1" align="center">
        <tr><td height="35" colspan="2"
bgcolor="#009933">&nbsp;  </td>
        </tr><tr>
            <td height="30" colspan="2" align="center">
```

```
<font color="#FF0000" size="+1"> 用户注册 </font></td>
</tr>
<tr>
  <td width="116" height="45"> 用户名 : </td>
  <td width="289"><input type="text"
name="username" size="19"/>
  <font color="#FF0000">*</font></td>
</tr>
  <tr>
    <td height="41"> 密码 : </td>
    <td><input type="password" name="password"
size="20">
    <font color="#FF0000">*</font></td>
  </tr> <tr>
```

```
<td height="41"> 密码确认 : </td>
    <td><input type="password" name="repassword"
size="20">
<font color="#FF0000">*</font></td>
    </tr>
<tr>
    <td height="46" colspan="2" align="center">
        <input type="submit" name="Submit" value=" 注
册 " />&nbsp;
        <input type="reset" name="reset" value=" 重置 " / >
    </td>
</tr><tr>
    <td height="35" colspan="2"
bgcolor="#009933">&nbsp;</td>
</table></form></body></html>
```

## 4.9.2 根据输入值自动计算的实例

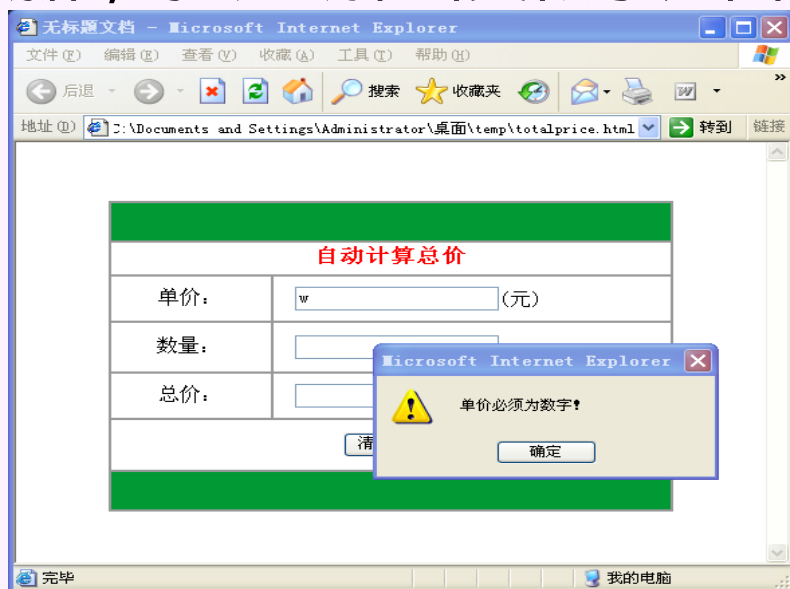
在一些外贸业务 Web 系统中，某些页面需要提供实时的辅助计算功能，例如：员工录入货物的单价和数量的值，通过 JavaScript 的事件处理可以直接显示出总价。如图 4-32 所示就是本例的运行效果图：



| 自动计算总价                             |                                         |
|------------------------------------|-----------------------------------------|
| 单价:                                | <input type="text" value="11.04"/> (元)  |
| 数量:                                | <input type="text" value="23"/>         |
| 总价:                                | <input type="text" value="253.92"/> (元) |
| <input type="button" value="清 空"/> |                                         |



本例中也采用了数字有效性验证，如果用户没有在文本框中输入合理的数据，系统会弹出类似于如图 4- 33 所示的警告对话框。



本例中定义了一个无参数的函数 `price_total()` 同时作为单价和数量的 `KeyUp` 事件处理程序。此外，我们还定义了一个含有两个参数的公共函数 `total ( price , amount )` 用来计算总价格。本实例的主要代码如下：

```
<html>
<head>
<script language="JavaScript">
function total(price,amount){
var totalprice=parseInt(amount)*parseFloat(price);
totalprice=Math.round(totalprice*100)/100;
document.form1.totalprice.value=totalprice;
}
function price_total(){
document.form1.totalprice.value="";
var amount=document.form1.amount.value;
var price=document.form1.price.value;
if(isNaN(price)){
alert(" 单价必须为数字！ ");
document.form1.price.focus();
```

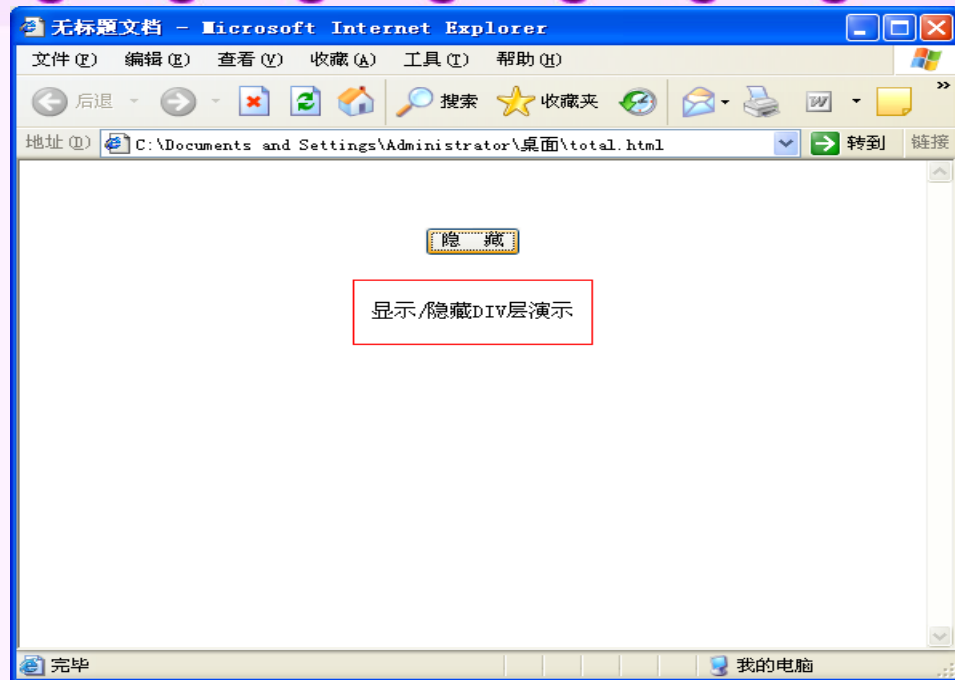
```
document.form1.price.select();
return false;}
if(isNaN(amount)){
alert(" 数量必须为数字！ ");
document.form1.amount.focus();
document.form1.amount.select();
return false;}
if(price!=""&&amount!="")
total(price,amount);}
</script></head><body>
<form action="" name="form1">
<table ><tr>
    <td > 单价： </td>
    <td><input type="text" name="price" size="20"
onKeyUp="price_total()" >( 元 )</td>
```

```
</tr>
  <tr>
    <td > 数量 : </td>
    <td ><input type="text" name="amount" size="20"
onKeyUp="price_total()" /></td>
  </tr>
  <tr>
    <td > 总价 : </td>
    <td><input type="text" name="totalprice"
size="20" readonly>( 元 ) </td></tr>
</table>
</form>
</body>
</html>
```

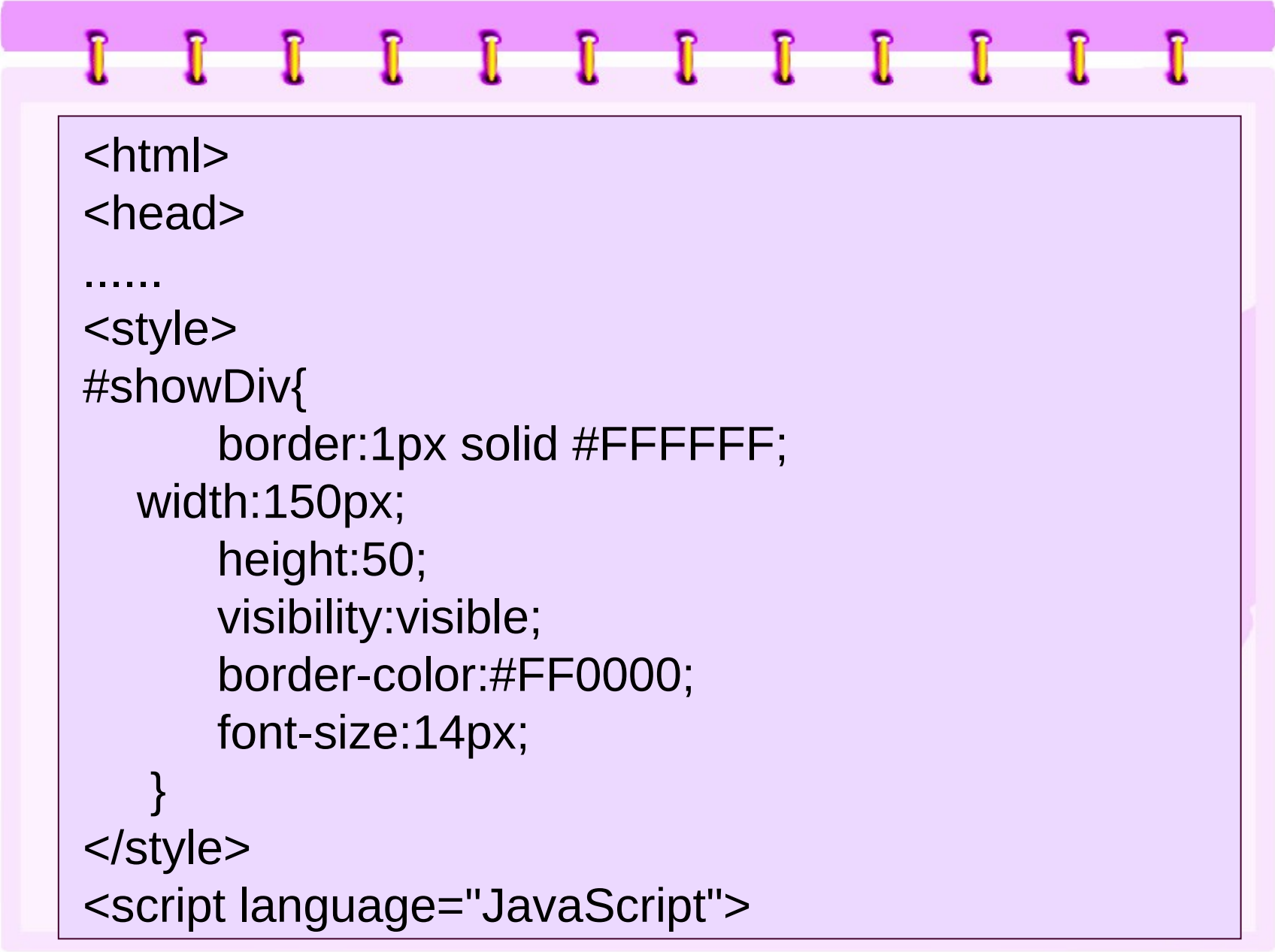
## 4.9.3 控制 Div 显示与隐藏的实例

对于显示和隐藏 Div，我们既可以用 display 属性，也可以用 visibility 属性，很多读者容易将 display 和 visibility 搞混淆，它们看似没有什么不同，其实差别却是非常大的。visibility 属性用来确定元素是显示还是隐藏，可以用 visibility="visible | hidden" 来表示，“visible”表示显示，“hidden”表示隐藏。当 visibility 被设置为“hidden”的时候，元素虽然被隐藏了，但它仍然占据它原来所在的位置。而 display 属性就有一点不同了，display 被设置为“none”时，元素实际上就从页面中被移走，它下面所在的元素就会自动跟上填充，所以它不会再占据原来所在的位置。

以下将采用一个 button 按钮作为控制开关，借助它的 onClick 事件来控制 Div 层的显示与隐藏。如图 4- 37 所示是本例的初始化页面，此时 Div 层处于显示状态。



单击【隐藏】按钮，我们可以发现，Div 层消失了，同时按钮上的文字变为“显示”，如果再次单击【显示】按钮，会发现页面又变回初始化页面，按钮上的内容也会随之改变，这就是将几种技术结合在一起使用的动态效果。



```
<html>
<head>
.....
<style>
#showDiv{
    border:1px solid #FFFFFF;
    width:150px;
    height:50;
    visibility:visible;
    border-color:#FF0000;
    font-size:14px;
}
</style>
<script language="JavaScript">
```

```
function hid_show(){
var obj=document.getElementById("showDiv")
if(obj.style.visibility=="hidden"){
obj.style.visibility="visible";
document.getElementById("button").value=" 隐 藏 ";
}
else{
obj.style.visibility="hidden";
document.getElementById("button").value=" 显 示 ";
}
}
</script>
</head>
<body>
```



.....

```
<input type="button" id="button" value=" 隐 藏 "
onClick="hid_show()"><br/><br/>
<Div align="center" id="showDiv">
<br/> 显示 / 隐藏 DIV 层演示
</Div>
</center>
</body>
</html>
```

## 4.10 本章小结

本章主要结合 Eclipse 最新的插件工具 Apatana 介绍了客户端动态脚本 JavaScript 的设计，其中重点介绍了 JavaScript 中基于对象的设计模式并通过实用的案例加以分析。JavaScript 脚本是 Web 设计中不可或缺的客户技术之一，也是本书后续高级开发中 Ajax 技术的设计基础，因此读者必须对 JavaScript 的开发技术熟练掌握。

# 本章习题

- 1、JavaScript 与 Java 语言有什么区别？
- 2、JavaScript 提供了哪些常见的浏览器对象？请举例说明。
- 3、JavaScript 事件驱动与处理的基本原理是什么？
- 4、请制作一个网页 index.html，按以下方式弹出新窗口，要求：
  - (1) 在浏览器加载完 index.html 后会自动弹出一个新窗口；
  - (2) 新窗口中显示外部 Internet 站点中的一幅图片；
  - (3) 新窗口的位置在屏幕左上角的位置。
- 5、请在第 3 章书后习题设计的表单页面中增加以下验证：
  - (1) 用户名、密码不可以为空；
  - (2) 密码长度不能少于 6 位；
  - (3) 两次输入的密码必须相等；

如果不满足以上的验证，则弹出警告框提醒用户进行正确的操作。

6. 请在网页中设计两个表格，要求采用 JavaScript 和 DIV 技术进行控制，通过按钮可以使两个表格的显示相互切换

